Deep Learning Wavefront Sensing

Via Raw Shack-Hartmann Images

J.M. Bekendam

Master Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Deep Learning Wavefront Sensing

Via Raw Shack-Hartmann Images

Master Thesis

J.M. Bekendam

June 18, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

The Delft Center for Systems and Control (DCSC) 'Smart Optics' aim to achieve higher resolution imaging through Adaptive Optics (AO). Adaptive optics is a modern technique for detecting and correcting real-time wavefront aberrations and is widely used in biomedical imaging and astronomical imaging. Wavefront sensing lies at the core of Adaptive Optics and is known to pose some challenges. Measurement of the wavefront cannot be done directly and has to be estimated through an intensity distribution on a detector. One approach to wavefront sensing is by using a Shack-Hartmann (SH) sensor. A Shack-Hartmann sensor (a pupil-plane sensor) subdivides the wavefront into N spatial areas using sub-apertures. The individual slopes across all sub-apertures are integrated to reconstruct the wavefront. The major advantage of using a Shack-Hartmann sensor is its fast operation speed, caused by the linear relationship between local slopes and original wavefront. This enables real-time wavefront reconstruction. The Shack-Hartmann sensor however, has some limitations. Its ability to reconstruct higher-order aberrations is restricted by the amount of lenses within the micro-lens array. Furthermore, a centroiding algorithm is used to compute the local slopes. Going from spots to centroids decreases the amount of informative pixels and greatly limits its wavefront reconstruction potential. Moreover, these centroiding algorithms often add a measure of uncertainty since spots can have irregular shapes or cross-over/overlap.

In this Master Thesis a novel approach to phase reconstruction from the raw SH measurement is proposed. Here, we show that Deep Learning techniques in combination with a micro-lens array can surpass traditional SH phase reconstruction methods and alleviate their current limitations. The proposed method uses the entire Shack-Hartmann Pattern (HP) as input to a neural network, supplying the network with more information than existing Deep Learning SHWR methods, which still rely on centroids. Using this approach, we can combine the accuracy of sensor-less techniques with the speed of a Shack-Hartmann sensor. Three different neural network architectures are considered in this thesis. Two of these neural networks (Alex-Net and Xception) are adapted to output a series of Zernike coefficients. Using these estimated Zernike coefficients, a wavefront can be reconstructed. The remaining neural network, U-Net, performs a direct pixel-wise estimation of the phase-map. The input Shack-Hartmann patterns are created using different micro-lens array (MLA) geometries, consisting of 25-, 256- or 900 lenses. The networks are evaluated on their ability to reconstruct a combination of 32- or 100- Zernike coefficients.

During this Master Thesis, we have reached our three set goals. Firstly, proving the viability of the Shack-Hartmann sensor as preconditioner inside a Deep Learning AO framework. The Shack-Hartmann sensor is still widely used in telescopes and microscopes and introducing a method of improved accuracy for these devices is of great interest to many fields of research. Secondly, we have compared the wavefront reconstruction accuracy for multiple state-of-the-art neural networks. Hereby showing that the Xception network excels in recognizing optical patterns, like raw Shack-Hartmann images, Lastly, we have shown that neural networks are less restricted by the amount of lenses within the MLA, in their ability to estimate a certain amount Zernike coefficients. Traditional wavefront reconstruction methods struggle to reliable estimate more Zernike coefficients than the used amount of lenses in the MLA (e.g. estimating 25+ Zernike coefficients using 25 lenses). Via our approach, a wavefront generated through a combination of the first 100-Zernike can reliably be reconstructed using a MLA consisting of 25-lenses, at least in simulation. Thus, the proposed tool not only relaxes the hardware restrictions, but also enables for higher-order Zernike estimation at high speeds.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to express my appreciation and gratitude to all the people I have collaborated with during this master thesis project.

First of all, I would like to thank Dr. Oleg Soloviev, my daily supervisor. His expert advice, enthusiasm, encouragement and support have been invaluable during my thesis. I would also like to thank professor Michel Verhaegen for his insightful comments and suggestions. Lastly, I would like to thank all members of the Smart Imaging Lab for all the interesting presentations and group meetings, which inspired me throughout the year.

Delft, University of Technology J.M. Bekendam
June 18, 2020

Chapter 1

Introduction

As light propagates from source to detector inside a homogeneous medium, it travels along a straight path. However, when light propagates into a turbulent media, its direction gets altered due to inhomogeneities. As a result, the shape of the wavefront is changed and no sharp focus is formed. Optical instruments suffer from these aberrations (distortions), since they deteriorating the quality of the image. To counter these aberrations, a control method has been developed (1950) called Adaptive Optics (AO, see Chapter 2). AO systems aim to minimize these abberations through a feed-back loop in which a corrective element reshapes the distorted wavefront back to its original shape [2]. The role of AO is first to acquire information about the distortions and then adjust a deformable mirror to compensate these optical aberrations. The phase profile of the wave is typically estimated with a wavefront sensor. The retrieved information is then passed to a control system which maintains the mirror shape as close as possible to the optimal configuration. A key component of AO systems is the measurement of the phase. This cannot be done directly due to the rapid oscillations of an Electro-Magnetic (EM) wave [6]. The wavefront phase has to be estimated from intensity distributions on a detector. This poses challenges since the relationship between intensity distribution on the detector and wavefront phase is non-linear. For this reason, multiple wavefront sensors and wavefront reconstruction techniques have been developed, in which a trade-off is made between reconstruction accuracy and operation speed.

The general approach to wavefront sensing is through image-based AO systems or wavefront sensor-based systems. Image-based AO systems are sensor-less and make use of a science camera and intensity-based techniques. Here, a specific image property is optimized, like Strehl ratio, to reconstruct the wavefront. Often, the sum of all pixel values is an appropriate optimization metric since the maximum value is related to zero aberrations [7]. Wavefront sensor-based systems often make use of a Shack-Hartmann sensor. A Shack-Hartmann sensor subdivides the wavefront using a micro-lens array (MLA) and creates a focal spot pattern. The first moment of this intensity distribution gives an approximation of the local averaged slopes. From these local slopes, reconstruction techniques can be applied to infer a wavefront [8]. A drawback of the Shack-Hartmann sensor is that its ability to reconstruct higher-order aberrations is inherently limited by the amount of lenses within the micro-lens array. Furthermore, due to the piece-wise linear estimation of the non-linear phase, high frequency content within the wavefront is unavoidably filtered. This causes

Shack-Hartmann based systems to have lower resolution capabilities than image-based AO systems. However, the linear property established between slopes and phase enables linear regression. As a result, very fast operation speeds are achieved which enable real-time wavefront reconstruction. Although image-based techniques have more accurate wavefront reconstruction capabilities, they are limited by their slow operation speeds. The slow operation speeds of these algorithms are due to non-linear relation between intensity image and wavefront phase. This renders them unable to perform real-time reconstruction ($\approx$ 10 ms) which is vital for certain instruments like telescopes. Atmospheric turbulence changes constantly and the shape of the mirror within telescopes is based on the correlation of each frame.

Machine learning, including Deep Learning (DL, see Chapter 3), have been introduced to the field of AO [9, 10] around 1990 and has been an ongoing field of research ever since. It was recognized that Artificial Neural Networks (ANN) could be used as an alternative to the traditional algorithms. ANN are information processing systems which are capable of learning complex non-linear input/output relationships, like intensity images to wavefront phase. Computational limitation caused that only shallow structured ANN could be used for AO pipeline, which deemed insufficient to completely remove the conventional algorithms. These shallow-structures have poor generalization and cause over-fitting. However, with the recent introduction of Convolutional Neural Networks (CNN, see section 3-2) to the field, significant improvements could be made. Due to the strong generalization capabilities of CNN's, DL methods are now able to compete with the traditional AO methods.

Regarding image-based AO systems, DL networks are often trained to map Point-Spread-Function (PSF, see Section A-1-5) to Zernike coefficients (see Section A-1-6) or directly to a phase-map [11, 12]. The drawback of these methods is that they require multiple images to infer a unique solution. In [13], preconditioners are proposed to enhance the amount of informative pixels, such that a single intensity image could be sufficient. Deep Learning Shack-Hartmann Wavefront Reconstruction (DL SHWR) on the other hand, has been utilized to map slope displacements to Zernike coefficients or directly to phase-map, at least for lower-order aberrations [14, 15]. The drawback of these methods is that a centroid algorithm is still required to compute the spot centers. Going from spots to centroids decreases the amount of informative pixels and greatly limits its wavefront reconstruction potential. Furthermore, these algorithms often add a layer of uncertainty since spots can have irregular shapes or cross-over and overlap.

In this Master Thesis a novel approach to phase reconstruction from the raw SH measurement is proposed. This approach involves state-of-the-art Deep Learning techniques. In [13], preconditioners (e.g. overexposure) are used within a DL sensor-less AO framework. These preconditioners increase the amount of informative pixels within the input images. We propose to use the micro-lens array as preconditioner for DL SHWR. Hence, our proposed method will use the entire Shack-Hartmann Pattern (HP) as input to a neural network, which should supply the network with more information than previously mentioned DL SHWR methods. The research question is formulated as:

- Is the micro-lens array as effective as the preconditioners mentioned in [13]?

Neural networks vary in size and complexity, and all search for patterns via different structures. Finding the optimal network structure is to some degree trail and error based. Hence, for this

thesis, three different state-of-the-art neural network architectures are considered: Xception, U-Net and Alex-Net. Xception and Alex-Net output a vector, containing the Zernike coefficients. U-Net however, is build for pixel-wise classification, such that it can directly output a phase-map. The next research question is formulated as:

- How do the considered neural networks compare in wavefront reconstruction accuracy and speed?

Traditional methods struggle to reliable estimate more Zernike indexes than the used amount of lenses in the MLA (e.g. estimating 25+ Zernike coefficients using 25 lenses). The corresponding research question is formulated as:

- Can neural networks reliably estimate more Zernike coefficients than the amount of lenses used in the MLA?

To answer these questions multiple Deep Learning models will be trained, using HP's emerging from different MLA geometries and wavefront aberrations.

1-1 Thesis Outline

The structure of this thesis is as follows.

Chapter 2 presents an introduction to the field of AO, which includes wavefront sensing, pupil-plane- and focal-plane sensors. Comparisons and trade-offs will be discussed regarding these sensors. The final section of this chapter is about Deep Learning Wavefront Sensing, which combines the two fields of interest, AO and DL. Chapter 3 is about Deep Learning, where each consecutive section takes the reader through the evolution of Neural Networks.

Chapter 4 outlines the first stage of our numerical experiments, generating data for the neural networks. To train a DL model, we need a vast amount of input/output data pairs. The more examples we show the model during training, the better it will adapt/generalize. Our input data will consist of HP and the output data will be the corresponding Zernike coefficients or phase map. The chapter starts with a general description of the proposed HP's. The different HP's emerge from two types of wavefront aberrations (32- or 100-Zernike polynomials) in combination with three MLA settings. These MLA settings differ in the total amount of lenses within the array. The consecutive sections indicate our method of simulating a HP within Python, taking noise and physical parameters into consideration. Chapter 5 outlines the second and final stage of our numerical experiments, using the data pairs to train and evaluate the proposed Deep Learning models. First, our neural network adaptations and training protocols are discussed. Then, the achieved results using the three neural networks (Xception, U-Net and Alex-Net) are reviewed. Finally, the best achieving model will be subjected to a noise robustness analysis. Chapter 6 is the final chapter, concluding on the project and discussing future work.

Chapter 2

Adaptive Optics

Adaptive optics is a modern technique for detecting and correcting wavefront aberrations and is widely used in biomedical imaging and astronomical imaging. It involves a wavefront sensor, providing essential information for reconstructing a wavefront. The reconstructed wavefront is used by an adaptive mirror to compensate for the induced aberrations. This thesis revolves around wavefront reconstruction. In sections 2-1 and 2-2, an introduction to AO and wavefront sensing is presented. The next two consecutive sections, 2-2-3 and 2-2-1, are about two essential sensors: focal-plane- and pupil-plane sensors. The former measures the wavefront in the focal-plane using a science camera, and is therefore known as a wavefront sensor-less method. In the last section, the pros and cons of each sensor will be discussed and the subject of Deep Learning Wavefront Sensing will be introduced.

2-1 Background

Horace Babcock was the first to propose the use of a deformable optical element in combination with a wavefront sensor, to correct for atmospheric aberrations (1953). Optical aberrations (i.e optical distortions) are harmful to an optical system and degrading the quality of the image. Figure 2-1 illustrates how much optical aberrations can affect the image formation and how effective Adaptive Optics can be to counter them.

Figure 2-1: Neptune with- and without Adaptive Optics. [1]

Since 1953, research has been ongoing into this topic, mainly focusing on telescopes and military development. In the twenty-first century, AO technology was introduced to microscopy and made its rise in the commercial- and biomedical sector. The introduction of AO in these different sectors yielded lower costs due to a rise in volume and simplifications. AO provides a method to enhance the performance of an optical system by controlling the wavefront. Optical systems demand a very high degree of precision since even the smallest distortions can have a large impact on the quality of an image. Distortions which influence the shape of a wavefront include temperature, atmospheric turbulence, mechanical actions or manufacturing errors in optical instruments.

2-1-1 The Principles of Adaptive Optics

Light traveling from source to detector can be represented as a wave, which depends on spatial coordinates $\mathbf{x} \in \mathbb{R}^2$ and time t. A wavefront is defined as a surface perpendicular to the energy propagation. For a source of monochromatic light, this is related to its complex amplitude

$$w(\mathbf{x}) = A(\mathbf{x})e^{-i\varphi(\mathbf{x})}, \tag{2-1}$$

where $A(\mathbf{x})$ is the amplitude and $\varphi(\mathbf{x})$ the phase. Often a paraxial approximation is considered for the wavefront, equating it to the phase distribution $\varphi(\mathbf{x})$. The deviation of the wavefront from its diffraction-limited form (flat or spherical) causes a decrease in resolution in the image formation.

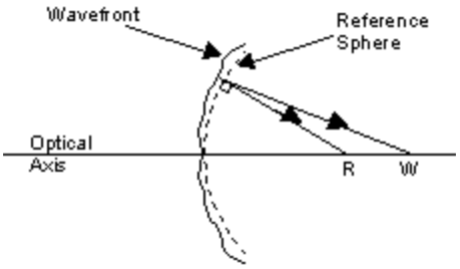


Figure 2-2: Illustration of a wavefront aberration. If the reference wavefront has a spherical shape (e.g. from a point source), the wavefront will form a perfect focus at the center of the sphere (R), and produce a perfect image. If the spherical wavefront is affected by optical aberrations, some of the rays will not pass through the center (W) and a blurry image is formed.

An AO systems can compensate for aberrations until the inherit diffraction-limited resolution of the optical system is reached. This resolution is defined by the Rayleigh resolution criterion (Section A-1-5). A wavefront from a distant star has an diffraction-limited flat wavefront (i.e. $\varphi(\mathbf{x}) = 0$) when it reaches the outer surface of the earths atmosphere. Inside the earths atmosphere, turbulence causes time and space varying Optical Path Length (OPL, Section A-1-2) differences in the wavefront. As a result, an aberrated wavefront enters the optical system (e.g. a telescope). The AO loop tries to counter these aberrations by introducing the exact opposite OPL, in an attempt to flatten the wave. Figure 2-3 shows the principle behind AO systems [16].

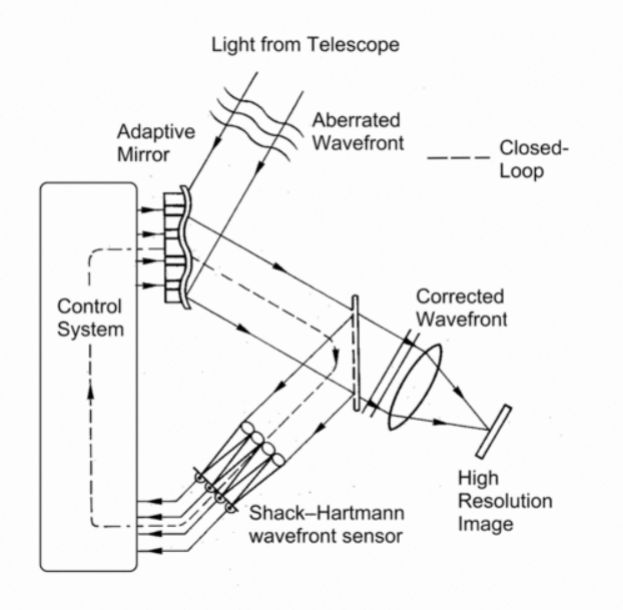


Figure 2-3: Principle of Adaptive Optics closed-loop system. In this schematic, we see a Shack-Hartmann sensor (Section 2-2-1), which is a type wavefront sensor. The Shack-Hartmann sensor consists of a micro-lens array and a detector (often a Charged Coupled Device). The micro-lens array subdivides the wavefront and as a consequence, multiple focused spots on the CCD are created. The position of each individual spot compared to a reference position and is used by the control system to approximate a wavefront. The adaptive mirror takes the form of the inverted estimated wavefront, such that the incoming wavefront is flattened. [17]

There are four main components to an AO system: the image detector, the adaptive mirror, the wavefront sensor and the control system. A beam splitter is placed in the optical path such that part of the incident wave is reflected onto the wavefront sensor, a Shack-Hartmann in Figure 2-3. The information obtained from this sensor is used by the control system to generate a control matrix which determines the shape of the adaptive mirror. The surface of this mirror is deformable such that it can influence the OPL of an EM-wave. After correction, the wave propagates to the image-detector, often a CCD, which will generate the final image.

2-1-2 Closed-Loop Control

In closed-loop AO systems, the wavefront sensor detects the wavefront after it has been corrected by the adaptive mirror. Figure 2-4 illustrates the block diagram of a closed-loop AO system. In this diagram, $\varphi_i(x, y, t)$ indicates the original incoming wavefront at spatial point (x, y), at time t. Often, this wavefront is not static and changes over time due to noise and turbulence. The correction calculated by the control system and wavefront corrector is denoted by $\varphi_c(x, y, t)$. As a result, the residual wavefront is calculated as:

$$\varphi_r(x, y, t) = \varphi_i(x, y, t) - \varphi_c(x, y, t) \tag{2-2}$$

In such an AO feedback system, the goal is to minimize the residual wavefront (see Section 2-2-1).

The effectiveness of an AO system is thus heavily dependent on the accuracy of the wavefront sensor and control algorithm.

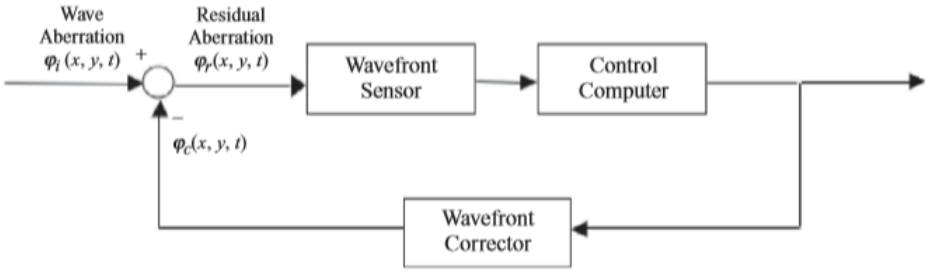


Figure 2-4: Block diagram of an simplified closed-loop AO system correcting an incoming wave aberration.

2-2 Wavefront Sensing

Wavefront sensing plays a key role in AO systems, it provides a method of (indirectly) measuring the shape of an optical wavefront. In closed-loop AO systems, this is used to measure how much a wavefront deviates from the diffraction limited case. The shape of deviated wavefront is used to generate a signal, which is sent to a corrective element. In AO-systems, one aims to minimize the error signal between these two wavefronts, diffraction limited and aberrated. Measurement of the phase of a light wave cannot be done directly [6]. The problem with directly measuring the phase is that optical sensors convert photons to an electric current. The electromagnetic field in the visual spectrum oscillates at frequencies ($\approx 10^{15}$ Hz) no electronic based optical device can follow. The general approach to wavefront sensing is through a focal-plane sensor or a pupil-plane sensor.

A focal-plane sensor (section 2-2-3) is a science camera (often a CCD) used for imaging. The relationship between recorded image and wavefront φ is given by the convolution between Point Spread Function (PSF, section A-2) and object intensity distribution. In such a sensor-less system a specific image quality metric is optimized, like image brightness or Strehl-ratio (see Section A-1-5), to reconstruct the wavefront. One advantage of image-based techniques is that the detected aberrations correspond to the aberrations affecting the image. This is not always true for wavefront sensor-based AO systems as different optical paths between imaging and sensing could be subjected to different aberrations (different optical properties/misalignment) [7].

Pupil-plane sensors (see Section 2-2-1) make use of a detector, such as a Shack-Hartmann sensor [2], located in a plane conjugated to the pupil-plane of the imaging instrument. A Shack-Hartmann sensor subdivides the wavefront and uses local gradients to approximate the original wavefront. The major advantage of sensor-based system is that only one measurement is required to infer a wavefront, whereas image-based systems need multiple. Single measurements is not only useful for telescopes, but also microscopes, where the to-be investigated sample would be less exposed to light. A drawback of pupil-plane sensors is their complexity and hardware costs.

2-2-1 Pupil-plane Sensor

The most widely used pupil-plane sensor is the Shack-Hartmann (SH) sensor [18]. A SH sensor subdivides the wavefront across N small spatial areas, using sub-apertures. As $N \longrightarrow \infty$ the wavefront can perfectly be represented. The aberrations at the location of each sub-aperture will be approximated as a tilt (i.e. a flat tilted wavefront). The individual tilts across all sub-apertures are integrated to reconstruct the full wavefront. This is done through either Zonal- or Modal reconstruction. Modal algorithms represent the wavefront by decomposition into a set of polynomials. The most widely used set for modal algorithms are the Zernike Polynomials (Section A-1-6) because they form an orthogonal set across circular apertures and because they are related to common optical aberrations [19]. Zonal algorithms build a phase field closest to the measured slopes using Least-Squares for example.

Using a lenslet array, a wavefront is sub-divided and every division is focused at different spots on a detection array behind it, often a CCD. This provides multiple slope measurements of the input wavefront.

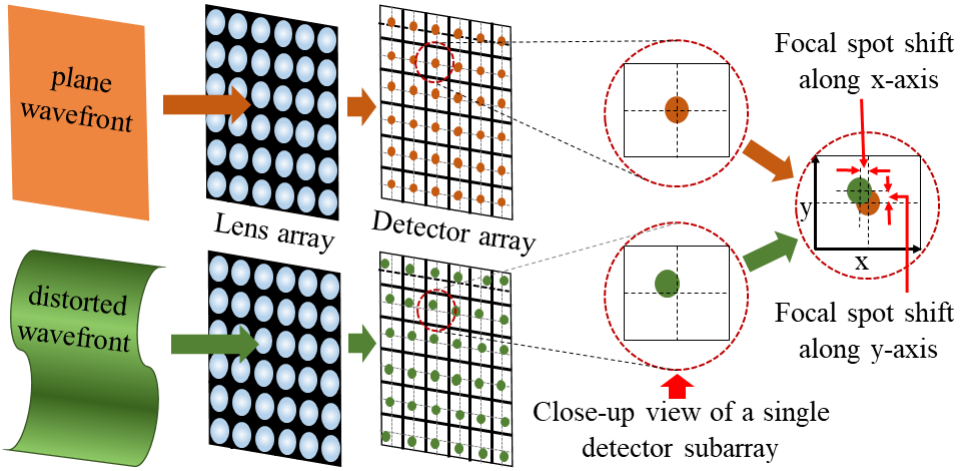


Figure 2-5: Schematic of the principle behind a SH sensor. A plane wavefronts cause unshifted focal spots on the detector array. Distorted wavefronts cause the focal-spots to shift depending on the local tilt. The change in focal spot locations is used to approximate a wavefront. [20]

When a plane wave goes through the lenslet array, a focused spot will appear in the centre of each sub-area of the detector array. If however, a distorted wavefront goes through the lenslet, the focused spots will be shifted in x- and y-directions. The reconstruction algorithms relate the change of focal spot centroid (i.e. first moment of intensity) to a local gradient at the center of a sub-aperture. This relation is build on the assumption that the part of the wavefront which goes into a sub-aperture is locally tilted. Although high-frequency content could be lost in this approximation, the linear property established between slopes and phase enables linear regression.

The Shack-Hartmann reconstruction process follows three basic steps: calculation of the centroid positions, conversion to wavefront slope, and wavefront reconstruction.

Centroid Positions

The location of each centroid is determined by the light intensity distribution on the CCD. With measured pixel intensity I_{ij}, the spot positions $x_{c,k}$ and $y_{c,k}$ are commonly determined by their first moments:

$$x_{c,k} = \frac{\sum_{i,j \in AOI_k} x_{i,j} I_{i,j}}{\sum_{i,j \in AOI_k} I_{i,j}} \text{ and } y_{c,k} = \frac{\sum_{i,j \in AOI_k} y_{i,j} I_{i,j}}{\sum_{i,j \in AOI_k} I_{i,j}}, \tag{2-3}$$

where k is the lenslet number and the summation is taken over AOI_k which are all pixels assigned to lenslet k. This centroiding process is one of the main contributors to a SH sensors inaccuracy. Often, it is not evident where the precise center of a spot is or which AOI_k a spot belongs to (see Figure 2-6).

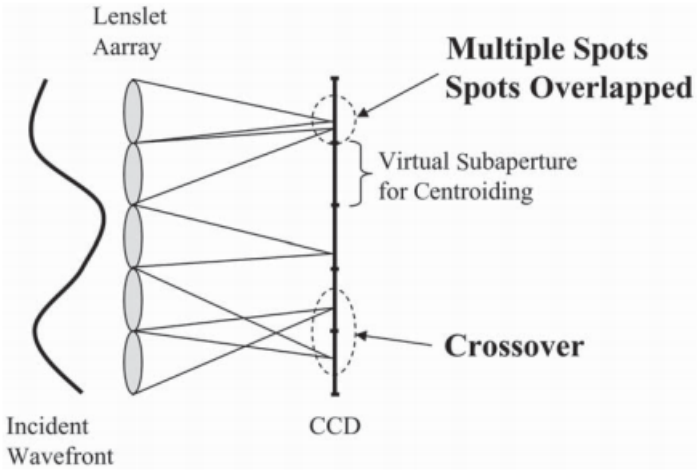


Figure 2-6: Limitations accuracy Shack-Hartmann sensor due to overlapping spots and spot-crossover.

The uncertainty in center spot location is due to the irregular shapes of the spots, which are enhanced by the discretization and noise (see Figure 2-7).

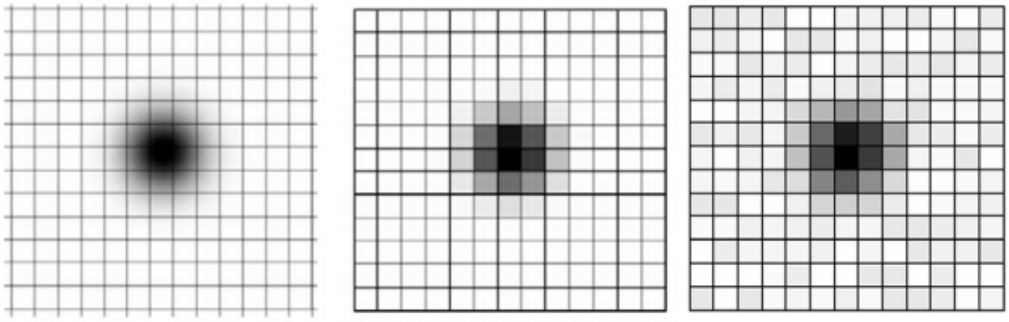


Figure 2-7: Spot on the detector array, (a) original, (b) discretized and quantized, (c) with noise. [21]

Wavefront slopes

The localized wavefront slopes are determined by the location of the wavefront slopes in comparison to a reference wavefront. For measured centroids $(x_c, y_c)_k$ and reference centroids $(x_r, y_r)_k$, the wavefront slopes are:

$$\begin{pmatrix} \langle \partial w/\partial x \rangle \\ \langle \partial w/\partial y \rangle \end{pmatrix}_k = \begin{pmatrix} s_x \\ s_y \end{pmatrix}_k \approx \frac{1}{L_H} \begin{pmatrix} x_c - x_r \\ y_c - y_r \end{pmatrix}_k + \begin{pmatrix} \eta_x \\ \eta_y \end{pmatrix}_k, \tag{2-4}$$

where L_H is the distance between the micro-lens array and the CCD, normally this is the micro-lens array focal length. The noise term is represented by η, which also includes the higher-order aberrations simplified by tilt.

Wavefront reconstruction

Having calculated the center locations and wavefront slopes $\mathbf{s}$, the wavefront can be reconstructed either via a zonal- or modal approach. Modal reconstruction will briefly be described in this section. Modal reconstruction approximates the wavefront W using a finite sum of known basis functions (e.g. Zernike polynomials Z). These basis functions are weighted by a set of coefficients $\mathbf{c}$ and used to minimize the difference between estimated wavefront $\mathbf{Zc}$ and unknown actual wavefront $\mathbf{W}$.

$$\mathbf{Zc} - \mathbf{W} \cong 0. \tag{2-5}$$

We can solve for $\mathbf{c}$ using the slope measurements $\mathbf{s}$

$$\mathbf{c} = \mathbf{Z}^{\dagger}\mathbf{s}, \tag{2-6}$$

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \cdots \\ c_J \end{bmatrix} = \begin{bmatrix} z'_{11} & z'_{12} & z'_{13} & \cdots & z'_{1(2K)} \\ z'_{21} & z'_{22} & z'_{23} & \cdots & z'_{2(2K)} \\ z'_{31} & z'_{32} & z'_{33} & \cdots & z'_{3(2K)} \\ & & \cdots & & \\ z'_{J1} & z'_{J2} & z'_{J3} & \cdots & z'_{J(2K)} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \cdots \\ s_{(2K)}, \end{bmatrix} \tag{2-7}$$

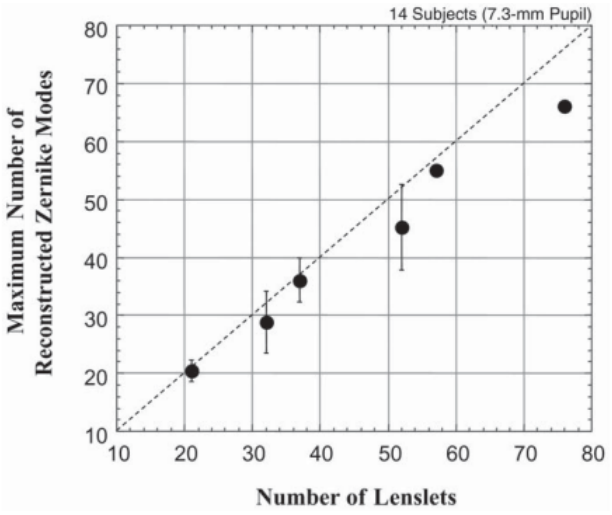
where z' of matrix **Z** are the derivatives of the basis functions, Z, the cross indicates the pseudo-inverse operation. The desired Zernike coefficients **c** are the result of the above multiplication. The index J denotes the amount of rows and indicates how many Zernike coefficients we want to recover. There are 2K columns for two times the amount of lenses used, since every lens yields a derivative with respect to x and y,

2-2-2 Performance Shack-Hartmann Sensor

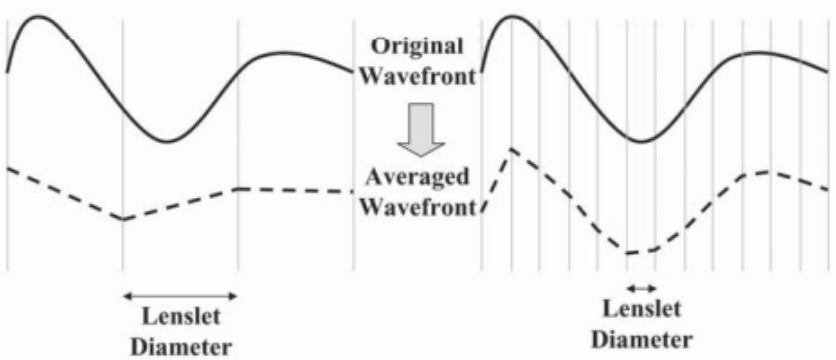
The performance of a Shack-Hartmann sensor is influenced by four main parameters: the number of lenslets inside the micro-lens array, the dynamic range, the sensor sensitivity and the MLA focal length.

Lenslets Versus Zernike Coefficients

As was be discussed in this Chapter, wavefront reconstruction can be done using Zernike polynomials (see A-1-6). Using these polynomials, the abberated wavefront consists of the first derivatives of individual Zernike polynomials, which respect to the x,y-displacements at each lenslet location. In general, Singular Value Decomposition (SVD) is then used to to calculate its inverse matrix. However, if the reconstruction algorithm attempts to calculate too many Polynomials with respect to the number of lenslets used inside the MLA, aliasing arises in the reconstructed wavefront [22].



(a) The maximum number of modes which can reliably be reconstructed given a number of sampling points (e.g lenslets). Dashed line represents a slope of 1. [22]



(b) Demonstration of different wavefront representations emerging from different amount of lenslets used. More lenslet yield a more precise wavefront.

Figure 2-8: Relationships between the number of lenslets used for reconstruction and reliable wavefront representation. [2]

As Figure 2-8 indicates, the minimum number of Shack-Hartmann spots required to fit a wavefront to a Zernike polynomial roughly equals to the number of Zernike modes to be fit.

Sensitivity

The sensor sensitivity (θ_{min}) and the dynamic range (θ_{max}) of the sensor bound the minimum and maximum phase detectable. Using the small angle approximation, we can define these parameters as:

$$\theta_{\min} = \frac{\delta y_{\min}}{f_{ML}}, \tag{2-8}$$

$$\theta_{\max} = \frac{\delta y_{\max}}{f_{ML}} = \frac{d/2}{f_{ML}}, \tag{2-9}$$

where θ_{min} is the smallest wavefront slope detectable and $\delta y_{\min}$ smallest spot displacement detectable. $\delta y_{\min}$ is determined by pixel size of the photo detector, accuracy of the centroiding algorithm and SNR ratio of the sensor. θ_{max} is the largest wavefront slope detectable. f_{ML} and d are the focal length and diameter of the micro-lens respectively. The trade-off between dynamic range and measurement sensitivity can be described as

$$\theta_{\min} = \frac{2\delta y_{\min}\theta_{\max}}{d}. \tag{2-10}$$

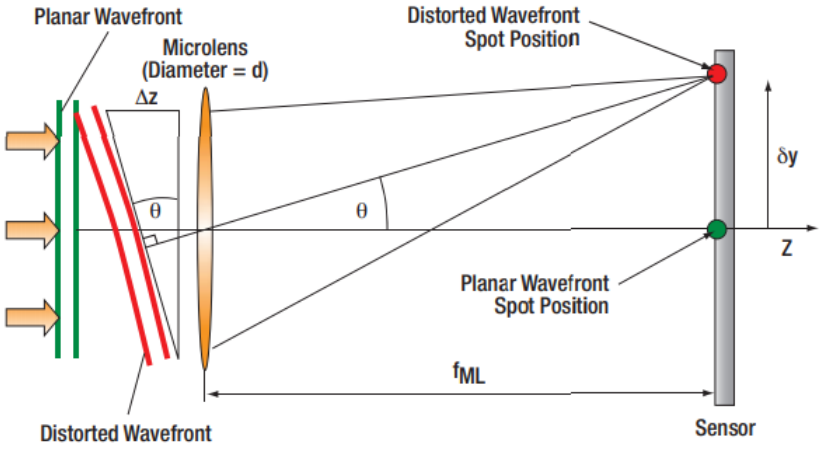


Figure 2-9: Imaging example through a single micro-lens. In case of a planar incident wave, parallel to the lenslet plane, a spot (green) will be created on the optical axis behind the lens. A distorted wavefront however, will cause a shift in spot (red) position along the x,y-axis with an angle θ. [22]

The minimum wavefront slope detectable depends on how accurate the spot displacements can be measured with respect to a reference position. This accuracy is limited due to the requirement of

a centroiding algorithm for the wavefront reconstruction process (see Section 2-2-1). The dynamic range can be increased by a shorter focal length or larger lenslet diameter. However, increasing the lenslet diameter yields a decrease in number of Zernike coefficients available. Furthermore, a shorter focal length decreases the sensor sensitivity.

2-2-3 Focal-Plane Sensor

Sensor-less wavefront sensing techniques use iterative algorithms to retrieve the wavefront phase φ from multiple intensity images [2]. Most common, these type of algorithms aim to retrieve the entrance pupil phase distortion through PSF's. A PSF is the impulse response of the optical system. The image of the focal plane can be written as the following:

$$I(x,y) = o(x,y) \otimes PSF(x,y), \tag{2-11}$$

where $I(x,y)$ is the image of the focal plane, $o(x,y)$ the ideal intensity distribution, $\otimes$ is the convolution operator. Based on Fourier imaging principles (see Section A-2), the PSF can be defined as:

$$\mathrm{PSF}(x,y) = |\mathcal{F}(w(x,y))|^2, \tag{2-12}$$

and,

$$w(x,y) = A_p(x,y)e^{j\varphi(x,y)}, \tag{2-13}$$

where $\mathcal{F}$ is the Fourier transform, A_p the transmittance function of the pupil and $\varphi(x,y)$ the wavefront phase over the entrance pupil.

Phase retrieval methods aim to finding this phase through model-based or model-free algorithms [23]. Model-free methods use a control signal as optimization variable and light intensity as performance function. Model-based methods aims to model a function based on the optical system to determine a input/output relationship.

Image-based AO systems have to deal with stagnation- and non-uniqueness problems when retrieving the wavefront from image information. As mentioned above, focal-plane sensors aim to retrieve the wavefront directly from an image of a blurred object. The non-uniqueness issue becomes evident when looking at the PSF formulation for a given wavefront $\phi(\alpha)$ and constant pupil-function A_p,

$$p(\phi) = \left|\mathcal{F}\left[A_p e^{j\phi}\right]\right|^2, \tag{2-14}$$

and an alternative wavefront $\phi' = -\phi(-\alpha)$,

$$p'(\phi) = \left|\mathcal{F}\left[A_p e^{j\phi'(\alpha)}\right]\right|^2 = \left|\mathcal{F}\left[A_p e^{-j\phi(-\alpha)}\right]\right|^2 = \left|\mathcal{F}\left[A_p e^{j\phi(\alpha)}\right]\right|^2. \tag{2-15}$$

Equation 2-14 and equation 2-15 show that multiple wavefronts yield the same image of the same object [24]. Therefore, alternative methods are required to solve this non-unique problem

formulation. A well known method is called phase-diversity, in which multiple images of the same object are recorded under equal imaging conditions but with a known additional wavefront aberration [25].

2-3 Sensor drawbacks and comparisons

As described in Section 2-2, the most widely used wavefront sensing techniques are image-based or gradient-based (Shack-Hartmann). The SH wavefront sensor does pose some issues. As wavefront abberations increase, the spots are imaged further from their local optical axis. The dynamic range is thus limited to aberrations which sub-images are placed within their own respective sub-array. Therefore, the dynamic range is limited by the size of the detector array. Another drawback is the precision required in alignment and calibration of the sensor. System aberrations generated by optical components (i.e abberations due to lenses, beam splitters or manufacturing errors in MLA) lead to shifts in spot positions, which may cause incorrect measurements. Furthermore, the resolution of the restored phase is limited by the centroiding algorithm (as discussed in 2-2-1) and the piece-wise linear estimation of the non-linear physical phase. The advantages of the SH-sensor are its wide dynamic range, high optical efficiency and operation speed. The linear relationship between the sensor output and corresponding wavefront enables real-time reconstruction.

The advantages of image-based techniques are low requirement for optical hardware, no special need for calibration and high resolution of the restored phase. However, focal-plane sensors rely heavily on computationally expensive algorithms and suffer from non-unique solutions. These expensive algorithms cause stagnation problems. The non-linear optimization routines suffer from a large number of local minima [26]. Inadequate initialization may cause stagnation in one of these local minima. Since these algorithms depend on ill-posed deconvolution, a regularization term is required to prevent noise amplifcation [27]. This leads to (often) unavoidable loss of information. This loss of information can only be retrieved using a priori information about the true image. This is problematic for certain applications where unknown phenomena are investigated. Due the speed of correction (which increases the temporal correction error) and the above mentioned drawbacks, focal-plane WFS is incapable of real-time WF reconstruction.

Table 2-1: Pros and cons pupil-plane- vs. focal-plane sensors. The + indicates a pro, the - indicates a con.

Quantity	Pupil-plane sensor	Focal-plane sensor
Operation Speed	+	-
Resolution Restored Phase	-	+
Optical Efficiency	+	-
Complexity Setup	-	+
Hardware Cost	-	+

With the introduction of Deep Learning to the field of AO some of the drawbacks mentioned in Table 2-1 might be negated. Deep Learning has is an efficient information processing system which can learn the complicated non-linear relationships between detector read-out and incident wavefront phase. This learning process might take several hours and a great deal of data but post training no iterative algorithm is required. As a result, higher operational speeds can be achieved for focal-plane sensors. Deep Learning might also offer pupil-plane sensors higher resolution capabilities

since linear estimations are no longer used. Furthermore, our work shows that a DL approach decreases hardware restrictions.

2-3-1 Deep Learning Wavefront Sensing

Conventional imaged-based wavefront sensing techniques require computationally expensive algorithms and their optimization routine depend on initial values which cause stagnation problems. This is not only time-consuming, but may also reduces the stability of the system. Wavefront sensor-based systems often rely on a Shack-Hartmann sensor, which capabilities are inherently limited by the number of lenslets and the centroiding algorithms. Deep learning offers an alternative to conventional methods. The idea of combining Adaptive Optics and Deep Learning was first to introduced in a 1990 paper [28] by Angel, applying a neural network to measure optical phase distortion caused by air turbulence. In 1993 neural networks were applied for the wavefront reconstruction for the Hubble Space Telescope [9]. At this time, the development of neural networks was still at an early stage and only one-hidden-layer models (Section 3-1) were used. These shallow models deemed insufficient for large input sizes, like PSF-type data. Due to the lack of convolutional filters, these models had poor generalization and caused overfitting.

With the introduction of Convolutional Neural Networks (CNN's, see Section 3-2), Deep Learning techniques have become more feasible within the field of AO. Recently, several studies successfully utilized and implemented Deep Learning techniques within image-based AO pipelines. [11, 12, 13]. In [11], Xu *et al.* used tested multiple state-of-the-art CNN's to map PSF-images to Zernike polynomials (4th-64th). In [12], Guo *et al.* used a CNN to map PSF-images directly to corresponding phase-map. Both [11, 12] employ methods which still require multiple PSF's as input. In [13], Nishizaki *et al.* present a method of estimating the first 32-Zernike through a single intensity image and showed feasibility for overexposed, defocused, or scattered images.

Regarding (Shack-Hartmann) sensor-based AO systems, neural networks have also been applied to boost AO performances. In [15], Guo *et al.* applied artificial neural networks (not convolutional) to predict Zernike indexes from centroid displacements. They showed that even simple neural networks outperform traditional methods, at least for low-order Zernike. More recently, [29], Li *et al.* applied a neural network to calculate the centroids in extreme situations. In [14], Swanson *et al.* propose using x- and y slopes to estimate a phase-map directly using a state-of-the-art CNN.

All of the previous mentioned Shack-Hartmann wavefront sensing methods are still reliant on the accuracy of the centroiding algorithm to estimate Zernike coefficients from the spot displacements. Using spot displacements is computationally more efficient than using raw SH images (since an entire SH image contains more data points than multiple spot displacements). However, going from spot to centroid decrease the amount of available information inside the input data. Furthermore, the centroiding algorithm introduces a level of uncertainty due to the possibility of irregular spot shapes and the crossover/overlapping of spots.

Recent state-of-the-art networks, like Xception or UNET, now allow for fast inference of complex images. Such networks are now able to learn from raw Shack-Hartmann patterns, supplying the networks with more information than using local slopes and negate the centroiding uncertainty. The irregular shapes and overlapping spots can now be used to our advantage since they will

related directly to the incident wavefront. Such a model would combine the best of two worlds, a wavefront reconstruction model with the operation speed of a Shack-Hartmann sensor and the resolution capabilities of sensor-less methods.

Chapter 3

Deep Learning

Artificial Intelligence (AI) is a field of research that studies methods to build intelligent programs and machines. Machine Learning is a subset of AI that uses algorithms (e.g. Deep Learning) to provide systems the ability to learn and improve from a training routines. In this chapter the subject of Deep Learning will be discussed. Section 3-1 is about Artificial Neural Networks (ANN). An ANN is a nonlinear, adaptive information processing system. This system contains multiple interconnected processing units (neurons) which can be trained to learn non-linear relationships between an input and output. ANN lie at the core of Deep Learning and have proven to be a very powerful tool for complex visual tasks and offer an alternative to conventional Machine Learning solutions. Then, two more recently developed architectures will be discussed, named the Convolutional Neural Network and Fully Convolutional Network. Sections 3-4-1,3-4-2 and 3-4-3 are about the three custom neural networks considered in this thesis.

3-1 Artificial Neural Networks

A neural network can be defined as a mathematical function f that takes in an input (X) and produces an output (Y)

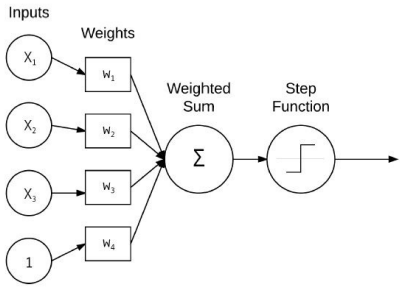
$$f : X \to Y. \tag{3-1}$$

This network function is composed of other functions $h(x, w)$ called neurons, which can be conveniently represented in a network structure (see Figure 3-1b). Neurons can be grouped to form a layer [30]. A single neuron takes a set of inputs x_i, computes the weighted sum, and produces a single output. The output of a single neurons is defined as:

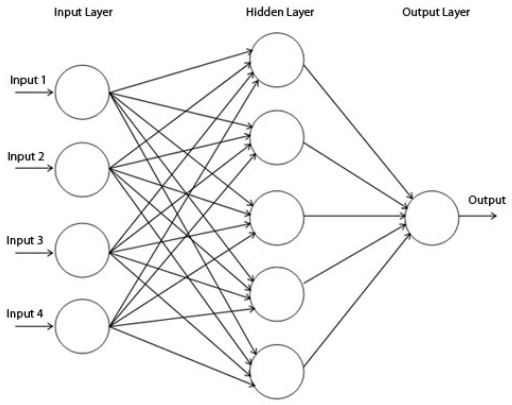
$$h(x, w) = \varphi\left(\sum_{i=0}^{N} w_i x_i\right), \tag{3-2}$$

where each input connection x_i is associated with a weight w_i. Furthermore, an activation function ϕ (see Figure 3-1a) is applied. Activation functions are used to ensure non-linearity's within the network and are further be discussed in Section 3-1-2. A single neuron could be used for simple linear binary classification. If the weighted sum exceeds a threshold, it outputs a positive class, otherwise a negative class (like a Logistic Regression Classifier or linear Support Vector Machine).

A neural network has three main sections: the input layer, hidden layers and an output layer (see Figure 3-1b) [30]. The input layer takes the input data and sends it to the first hidden layer. A network consists of one or multiple hidden layer, more hidden layers results a 'deeper' network. The output layer processes the information passed on from the last hidden layer to produce a result. Lets consider a classic Deep Learning example, training a neural network to recognize hand written digits. Training a network (see Section 3-1-1) means supplying the network with inputs (e.g. digitized hand written number 9) such that it can find patterns belonging to a specific output (e.g. the value 9). Let us consider two-dimensional images of size 10x10 pixels as input, containing the digitized hand written digits. Consequently, the input layer will have 10x10 = 100 neurons. The optimal amount of hidden layers (and their size) is never immediately evident and is found through cross-validation. With respect to the written digits, let us consider two hidden layers. We could expect the first hidden layer to search for loopy patterns or lines, and the second layer to search for certain edges. When the input data contains the handwritten digit '9', the hidden layers will 'search' for a combination of a straight line pattern and a circle pattern. The output layer will have 10 neurons, such that the network can output values from 0 to 9.



(a) Architecture of a single neuron using a step function as activation function.



(b) Architecture of fully connected neural network with multiple neurons in hidden layer.

Figure 3-1: Schematic of TLU- and neural network architecture. [3]

3-1-1 Training

Training a neural network is the process of finding the optimal network function f, by tuning the weights w, such that a desired output is found. Most neural networks are trained using a method called backpropagation [31]. The algorithm goes through the network in two passes (forward and backward) and computes the gradient of the network error with regards to every single parameter (or weight). Once it has all the gradients, it performs a Gradient Descent [32] (or a variation off

Gradient Descent). This process repeats itself until the neural network converges to a solution.

Going through the algorithm in more detail:

- During training not all data is fed into the network at once. It works in mini-batches and goes through the training set multiple times. Each pass through the network is called an epoch.
- During a forward pass, the algorithm computes the outputs of the hidden layers until the output layer is reached.
- The algorithm measures the output error by comparing it to the desired output (using a loss function).
- Then the contribution of each hidden layer output connection, to the error, is determined.
- Finally, Gradient Descent is used to optimize the weights.

The performance of a neural network is determined by a given loss function $L(\theta)$. This loss function depends on the network parameters (θ), original labeled output y, and is computed over batches of data during each epoch:

$$L(\theta) = \frac{1}{B}\sum_{b=1}^{N} \ell\left(\boldsymbol{y}, f\left(\boldsymbol{x}_b; \boldsymbol{\theta}\right)\right), \tag{3-3}$$

where $f\left(\boldsymbol{x}_b; \boldsymbol{\theta}\right)$ is the networks predicted output ($y_{predicted}$) and x_b represents the input data [30]. During training, the network tries to minimize this loss function $L(\theta)$ in an attempt to find the optimal system parameters that yield the closed relationship between y and $y_{predicted}$. The minimization process is performed by (or a variation off) batch gradient descent

$$\begin{aligned} \nabla L(\theta) &= \frac{1}{B}\sum_{b=1}^{N} \nabla_\theta \ell\left(\boldsymbol{y}_b, f\left(\boldsymbol{x}_b; \theta\right)\right), \\ \theta_{t+1} &= \theta_t - \alpha \nabla L(\theta), \end{aligned} \tag{3-4}$$

where α presents the learning rate [30]. This determines the step size of the gradient descent. If the step size is to large, it will have trouble converging to an minimum. If the step size is too small, the optimization routine will take a very long time (see Figure 3-7).

Figure 3-2: Illustrations of a 2D loss function $J(\theta)$ and the minimization process. Each plot shows iterations using different learning rates (α). A small α requires many iterations before reaching the minimum point. The optimal α rapidly reaches the minimum point using minimal iterations. To high an α causes drastic updates which lead to divergent behaviours. [33]

However, most optimization surfaces vary a lot along all directions, hence a constant learning rate often yields poor performance. To tackle the learning rate issue, stochastic gradient descent methods have been proposed like: Adaptive Moment Estimation (ADAM) or RMSProp [34, 35].

3-1-2 Activation function

Activation functions introduce non-linearity's to the neural network. Introducing non-linearity ensures the model is capable of learning more complex problems. If there is no activation function the output signal would just be simple linear function. This is because a chain of linear functions is still a linear function. It would only be capable of Linear Regression which is limited in potential if you want to analyse images or videos.

Common activation functions are: sigmoid, Rectified Linear Unit function 'ReLU' or soft ReLU. The sigmoid activation function is defined as

$$\phi_{\text{sigmoid}}(x) = \frac{1}{1+\exp(-x)}. \tag{3-5}$$

This function clips all input values between (0,1), this makes it particularly useful in the output layer in networks where probabilities are predicted. However, the sigmoid activation function is not very well suited for the hidden layers. This is because during backpropagation, derivative's are calculated with respect to the activation functions (see Equation 3-4). Inside the sigmoid, values close to 0 or 1 have a gradient close to zero. This is known as the 'vanishing gradients' problem, and causes the optimization routine to get stuck. In order to prevent these vanishing gradients, the ReLU activation function was introduced

$$\phi_{ReLU}(x) = \max\{0, x\}. \tag{3-6}$$

The ReLU prevents vanishing gradients in the positive regions, but could still cause saturated neurons in the negative regions. One solution to this is the 'soft ReLU', allow a small fraction in the negative part. Regular ReLU however, is computationally more efficient and helps to generalize.

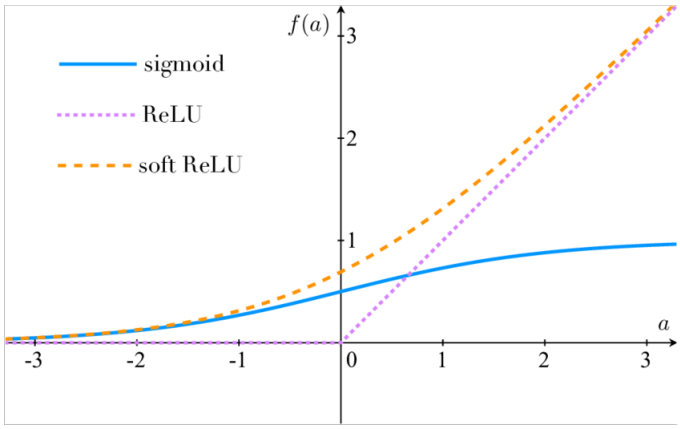


Figure 3-3: Three different activation functions. The sigmoid activation is often used in the output layer of a network, when a single probability has to be predicted. (soft) ReLU are the most widely used activation functions and are less prone to vanishing gradients.

3-2 Convolutional Neural Network

Deep neural networks (DNN) show very good results on linear regression- and very simple image recognition problems. However, they are not suited for more complex visual tasks like classification of detailed images, object detection (classifying multiple objects in an image) or semantic segmentation (classifying each pixel). For these complex tasks, convolution layers are required, creating Convolutional Neural Networks (CNN). Convolutional layers use filters to convolve patches of adjacent pixels, not only to reduce the operations needed to process the image but also because adjacent pixels together are often meaningful.

The most essential blocks of a CNN are its convolutional layers. Neurons in the first convolutional layer are not connected to every pixel of the input image (like discussed in the previous section) but only to the pixels in their receptive fields. This allows the network to concentrate on low-level features (shapes, e.g. a circle) in the first layers, then assemble them into high-level features (edges, lines) in the next layers. A neurons weights can be represented as an image the size of its receptive field. A set of weights are called filters, or convolutional kernels. When an image goes through a kernel, it outputs a feature map, which highlights the areas in an image that activate the filter the most. During training, these filters will learn to take on the most useful values for its specific task, and the layers above learn to combine them into more complex patterns.

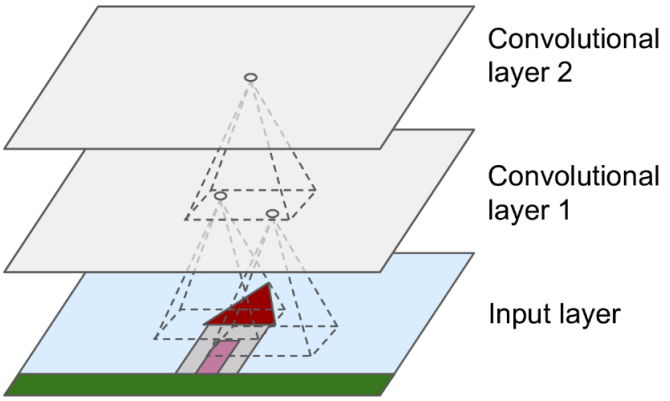


Figure 3-4: Convolutional Neural Network showing the receptive fields for training. Post training, when an image is submitted (e.g. of a house), each convolutional layer will output multiple feature maps. These feature maps highlight the parts the conventional layer was trained to look for. Feature maps resulting from the first convolutional layer would most likely show the outline of the house, roof or door. The second convolutional layer would output feature maps illustrating edges and lines of the house. [36]

All neurons within a feature map share identical parameters (i.e weights and bias). This not only reduces the number of parameters in the model but also ensures that a learned pattern is invariant of its location. This is in contrast to regular DNN, if it recognizes a pattern in one location, it will only recognize it in that particular location.

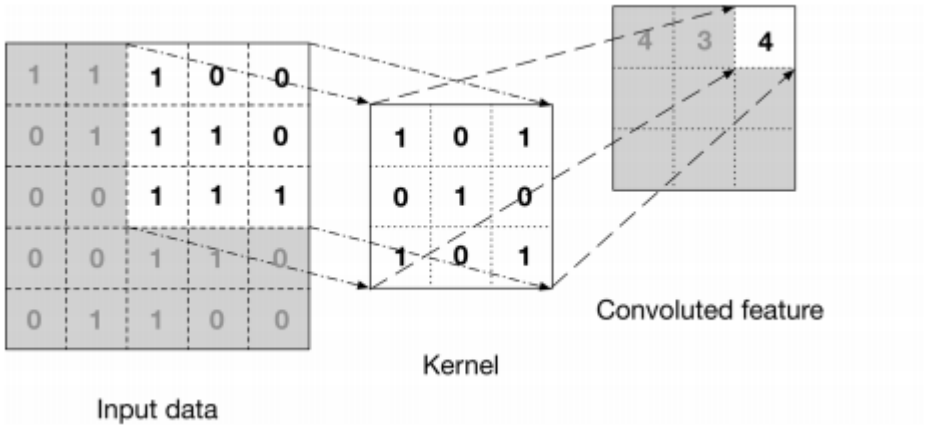


Figure 3-5: A Convolutional operation for input image $x \in \mathbb{R}^{1\times5\times5}$ and a convolutional kernel $u \in \mathbb{R}^{1\times3\times3}$. The output is a feature map, highlighting relevant features within the input image. [4]

Consider the input data $\mathbf{x}$ of size $\mathbb{R}^{C\times H\times W}$ and convolutional kernel $\mathbf{u}$ of size $\mathbb{R}^{C\times h\times w}$. A convolution operation is performed by sliding the kernel across the input image, and summing the element-wise product between the overlapping elements (see Figure 3-5).

$$\boldsymbol{o}_{i,j} = \boldsymbol{b}_{i,j} + \sum_{c=0}^{C-1} (\boldsymbol{u}_c * \boldsymbol{x}_c)[i,j] = \boldsymbol{b}_{i,j} + \sum_{c=0}^{C-1} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \boldsymbol{u}_{c,n,m} \boldsymbol{x}_{c,n+i,m+j} \tag{3-7}$$

The output **o** is of size $\mathbb{R}^{C\times(H-h+1)\times(W-\omega+1)}$ and is used as input to subsequent convolutional- or fully-connected layers.

3-2-1 Pooling layers

Training a convolutional neural network can be a time consuming process since it involves many iterations of computationally expensive operations. To reduce the training time, pooling layers were introduced. Pooling layers have one purpose: down-sample the feature map. By reducing the dimensions of the feature map, computational load is decreased while maintaining spatial information and decreasing over-fitting effects. Suppose an input tensor $\boldsymbol{x} \in \mathbb{R}^{C\times(rh)\times(mw)}$ is passed to a pooling layer of size h x w, then the max pooling operation can be defined as:

$$\boldsymbol{o}_{i,j} = \max_{n<h,m<w} \boldsymbol{x}_{c,ri+n,sj+m} \tag{3-8}$$

The output o has dimensionality $\mathbb{R}^{C\times r\times s}$ and is passed on the subsequent convolutional- or fully-connected layers.

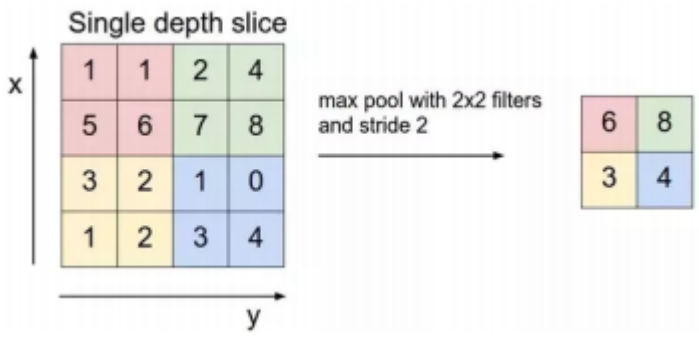
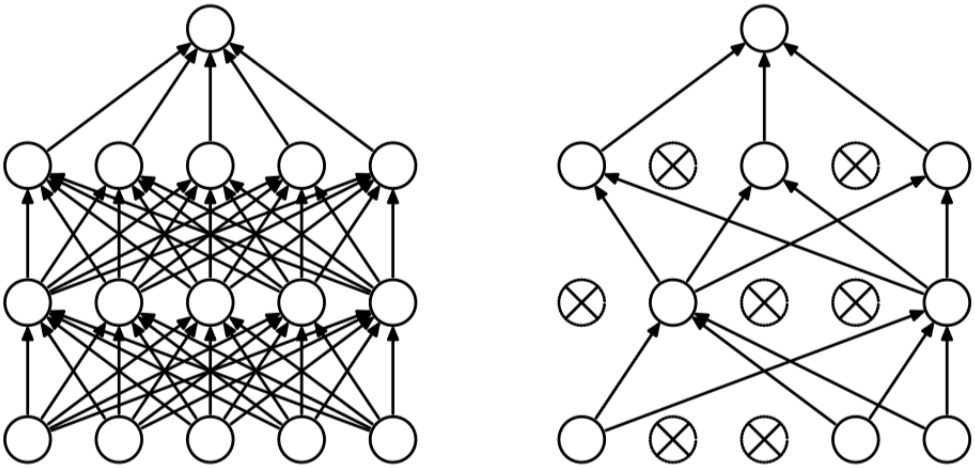


Figure 3-6: Example of a max-pooling operation with kernel 2x2 and stride of 2. The stride of a kernel determines the pixel step size over the image. A stride of 2 makes the kernel slide 2 pixels each iteration. [5]

3-2-2 Dropout layers

A typical regularization method is applying dropout on fully connected layers within the network. Regularization methods prevent overfitting caused by optimization. A model which has been overfitted performs well on the training set, but poorly on the test set. In essence it has failed to generalize. Dropout prevents this and works as follows: in each forward pass, a random set of neurons will be set to zero. This forces the network to be less reliant on certain combination of neurons. As a result, the model becomes more generalized.



(a) Standard Neural Net (b) After applying dropout.

Figure 3-7: Illustration of dropout within a neural network. During training, after each epoch, some neurons will be set to zero. This has proven to be an effective regularization method and forces the network to generalize. [37]

3-3 Fully Convolutional Network

FCN's were first introduced in [38] by Jonathan Long for semantic segmentation. Semantic segmentation is the classification of each pixel in an image according to the object or class it belongs too (see Figure 3-8). The proposed strategy is to replace the fully connected layers at the top of a Neural Network with convolutional layers. A major advantages of this strategy is that there are not restrictions on the input size of an image and it offers an efficient method for pixel-wise classification for the precise localization of objects.

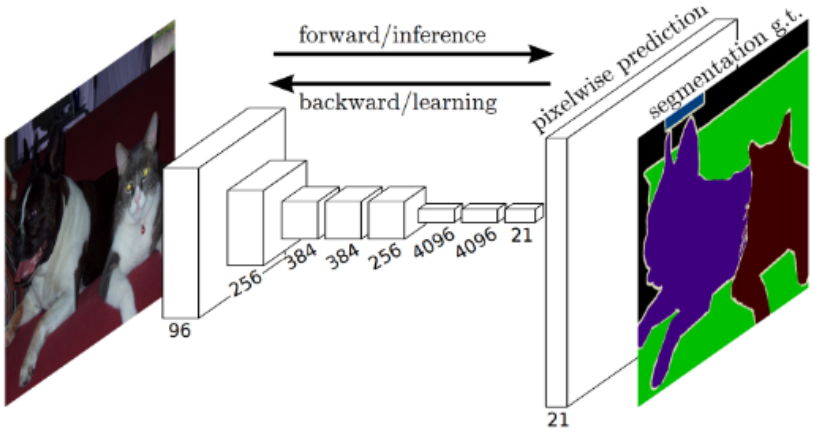


Figure 3-8: Fully Convolution Neural Network for pixel-wise classification. Note that the top fully connected layers have been replaced with convolutional layers. The values underneath the blocks indicate the amount of channels within the convolutional kernel. First, the image is gradually down-sampled, not unlike regular convolutional neural networks. Secondly, unlike CNN's, the image is up-sampled back to its original size. This yields a pixel-to-pixel classification of the original image. [38]

Conventional CNN's only down-sample the input image in its convolutional layers, FCN's however use up-sampling of the image in its top layers using transpose convolution. In [38], skip connections between different parts of the network are proposed. Complex or deep features can be found using a 'deep' network. However, spatial information is lost in these deeper layers. Shallower layers contain more local information. The result is greatly enhanced when these two information sources are combined using these skip connections.

As mentioned above, FCN enabled efficient pixel-wise classification which is a useful concept in the field of DLWFS. Input images such as PSF or SH-patterns can now be trained to directly output a phase-map, hereby avoiding Zernike calculations. This negate fitting errors Zernike polynomials may introduce.

3-4 Custom models

Many teams compete each year to win the ILSVRC challenge, this is an image classification competition. AlexNET was the first team to win using a CNN in 2012. From this competition a couple of fundamental architectures have emerged, like GoogLeNet (2014) and ResNet(2015). Figure 3-9 shows a couple of common models, comparing accuracy with complexity. The key difference between these networks are its networks size (a larger network has more parameters, hence training and inference time is greater) and accuracy. When choosing a network, the trade-off between inference (and training-) time and accuracy needs to be considered.

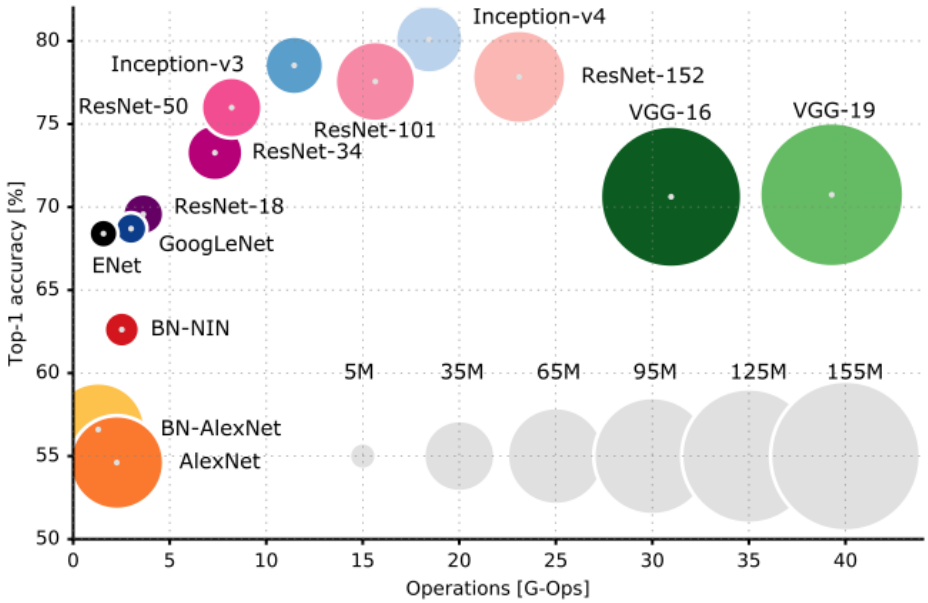


Figure 3-9: Top-1 accuracy vs. the number of operations required for a single forward pass in multiple popular neural network architectures. As can be deduced from this plot, more operations does necessarily mean more accurate estimations.

These fundamental architectures can be trained from scratch, starting with random initialized weights, but can also be used for transfer learning. Transfer learning is a method where a model developed for a task is reused as the starting point for a model on a second task. The transfer of previous knowledge from the first model to the second is often beneficial. This concept is especially powerfull when there is insufficient data. The interested reader can refer to [39] for more information on transfer learning.

3-4-1 U-Net

A drawback of Neural Networks is the need for a large amount of data for the training process, which is beyond the scope for certain applications. O.Ronnerberger's team provided a new type of FCN architecture, called U-Net, in [40], which greatly reduced the amount of required data and improved the inference speed for pixel-wise classification.

The major modification made to the standard FCN architecture is that the up-sample parts also have multiple feature channels. This way, local information can propagate to higher resolution layers. The up-sample part of the network is now symmetric creating an u-shaped architecture, hence the name of the network (see Figure 3-10).

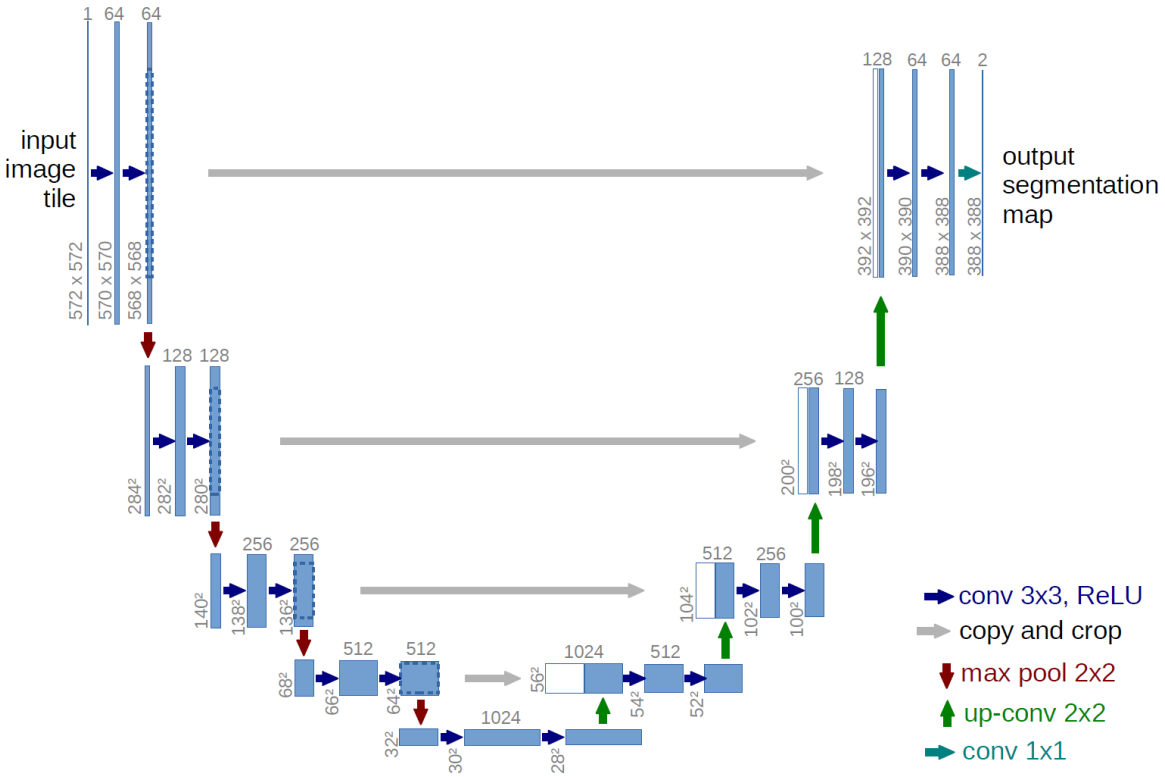


Figure 3-10: U-Net architecture, expensive path has multiple feature channels. The blue arrows pointing to the right indicate convolutional layers with the activation layer 'ReLU'. For 3x3 convolution, one pixel border is lost in the image. The downward path consists of pooling layers, which decreases the x,y size of the feature map by a factor of 2. After each pooling step, in the downwards path, the amount of feature channels are increased by a factor of 2 (these are the numbers on top of each layer). The downwards 'contracting' path gradually increases the width and decreases the height of the layers. The upwards 'expansion' path consists of up-convolution (increasing the output resolution) and concatenation with high-resolution features from the contracting path. This concatenation helps with localization of high-resolution features. A successive convolution layer can then learn to assemble a more precise output based on this information. [40]

3-4-2 Xception

Xception [41] is a state-of-the-art deep neural network (see Figure 3-11) which is based on the Inception-V4 network, which merges ideas from GoogleNet [42] and ResNet [43]. The novel approach of Xception is using Depth wise Separable Convolution Layers (DSEP) instead of convolution layers. These type of layers have two major advantages:

- Fewer parameters to tune, which reduces overfitting.
- Computationally cheaper due less operations.

Where standard convolutional layers try to simultaneously capture spatial patterns (e.g., a circle) and cross-channel patterns (e.g., circle + circle = number 8), separable convolutional layer assume these can be modeled separately.

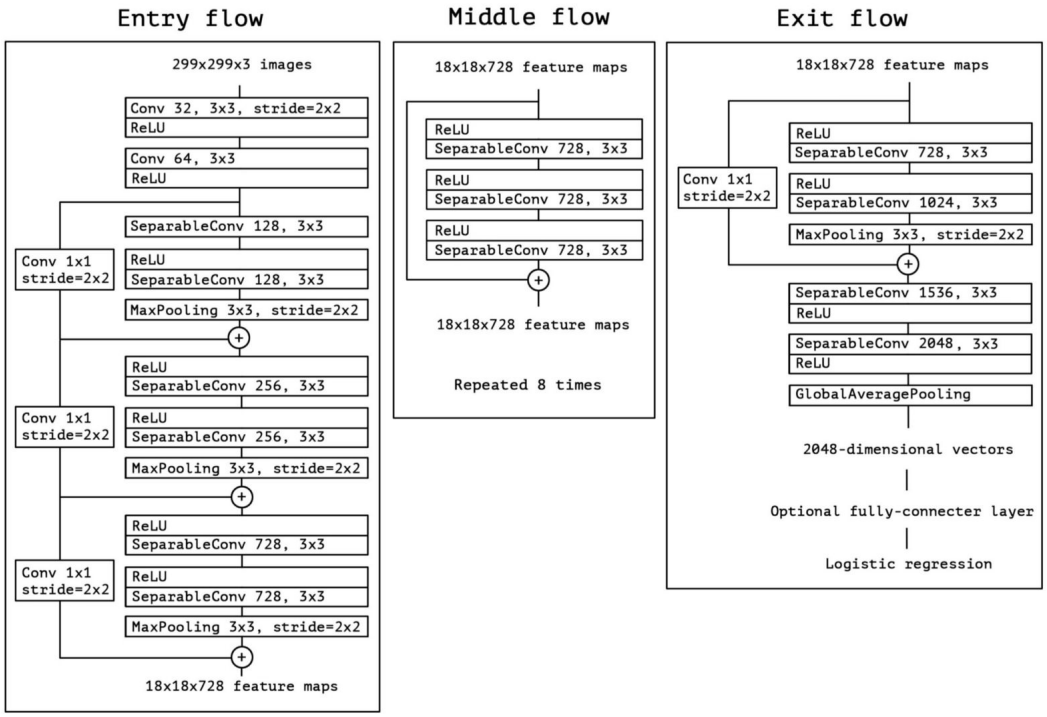


Figure 3-11: Xception Deep Neural Network architecture. The connections going around the Separable Convolutional layers are called residual blocks. These were introduced by GoogleNet and ResNet. These blocks allow for better flow of information to the last fully-connected layers and may prevent vanishing gradients during back propagation. The main activation function used in this network is ReLU and Max Pooling layers are used to down-sample the image. [41]

Depth wise Separable Convolution Layers

Suppose an input image of size $D_f \cdot D_f \cdot M$ (see Figure 3-12) and a convolutional layer with N filters of size $D_k \cdot D_k \cdot M$. There are N filters which slide over the image $D_p \cdot D_p$ times, hence the

output will be of size $D_p \cdot D_p \cdot N$. The kernel parameter D_p depends on the step size (stride) of the kernel over the image.

$$\text{number of operations per single convolution (C)} = D_k \cdot D_k \cdot M \tag{3-9}$$

$$\textbf{total amount of multiplications} = N \cdot Dp \cdot Dp \cdot C = N \cdot D_p^2 \cdot D_k^2 \cdot M \tag{3-10}$$

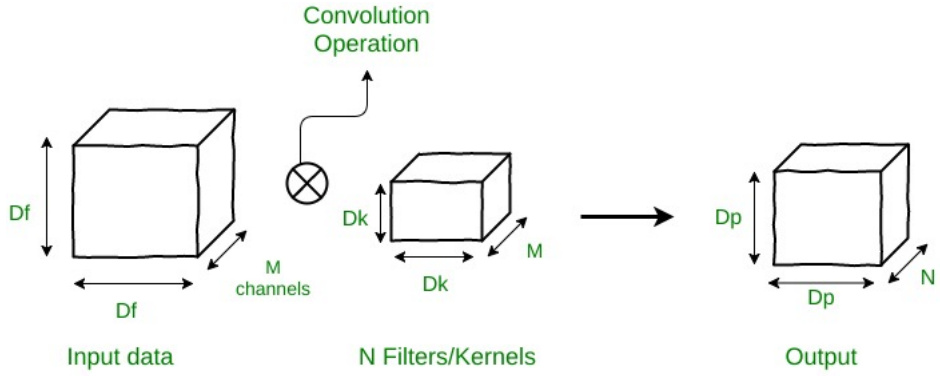


Figure 3-12: Standard operation of a convolutional layer. The input is convolved with N kernels, resulting a down sampled output. [44]

Equation 3-10 denotes the total amount of operations (multiplications) required in a standard convolutional layer. Lets consider operations performed by the DSEP layers, which are performed in two steps: Depth-wise convolutions and Point-wise convolutions. Depth-wise convolutions (see Figure 3-13) are applied only to a single channel, unlike normal Convolutional layers. Filters will be of size $D_k \cdot D_k \cdot 1$. Input data of M channels requires M such filters and the output size will be $D_p \cdot D_p \cdot M$.

$$\text{number of operations per single convolution (C)} = D_k \cdot D_k \tag{3-11}$$

$$\textbf{total amount of multiplications} = M \cdot Dp \cdot Dp \cdot C = M \cdot D_p^2 \cdot D_k^2 \tag{3-12}$$

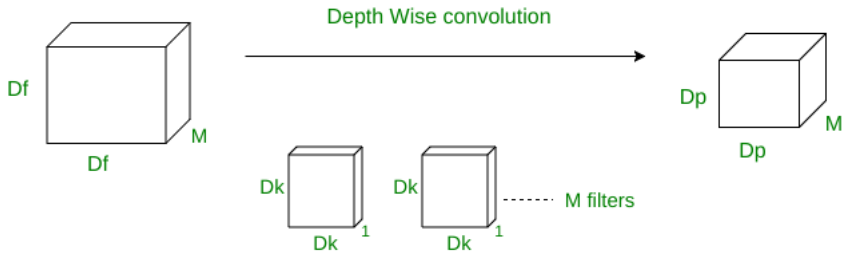


Figure 3-13: First step of a DSEP layer, Depth-wise convolutions. These Depth-wise convolutions use multiple filters with only one channel. [44]

The second step is the point-wise operation (see Figure 3-14), a $1 \cdot 1$ convolution is performed for each of the M channels. Hence, the filters will be of size $1 \cdot 1 \cdot M$ and the output size with N filters will be $D_p \cdot D_p \cdot N$. Note that this is the same output as the standard convolutional layer.

$$\text{number of operations per single convolution (C)} = 1 \cdot M \tag{3-13}$$

$$\textbf{total amount of multiplications} = C \cdot D_p \cdot D_p \cdot N = M \cdot D_p^2 \cdot N \tag{3-14}$$

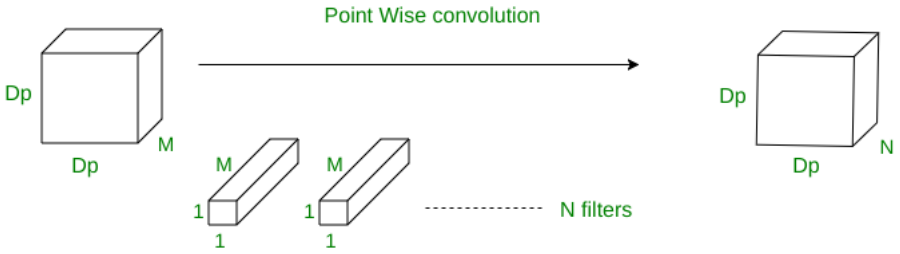


Figure 3-14: Second step of a DSEP layer, Point-wise convolutions. These Point-wise convolutions use multiple filters (N) of size 1 x 1 x M. [44]

Summing the operations performed in both steps will yield the total number of operations applied by the DSEP layers.

$$\text{Number of operations Depth Wise Separable} = M \cdot D_p^2 \cdot (D_k^2 + N) \tag{3-15}$$

Comparing Equation 3-15 with Equation 3-16 reveals DSEP layers substantially decrease the total amount of performed operations compared to standard convolutional layers.

$$\text{Number of operations standard convolution} = N \cdot D_p^2 \cdot D_k^2 \cdot M \tag{3-16}$$

3-4-3 Alex-Net

Alex-Net [45], created by Alex Krizhevsky, won the 2012 ImageNet Large Scale Visual Recognition Challenge in 2012, beating the other systems by a large margin. Alex-Net contains less convolutional operations than Xception or UNET and as a result, achieves faster training- and inference times respectively. However, less convolutional operations also limits the pattern recognition potential. The architecture is denoted in Figure 3-15.

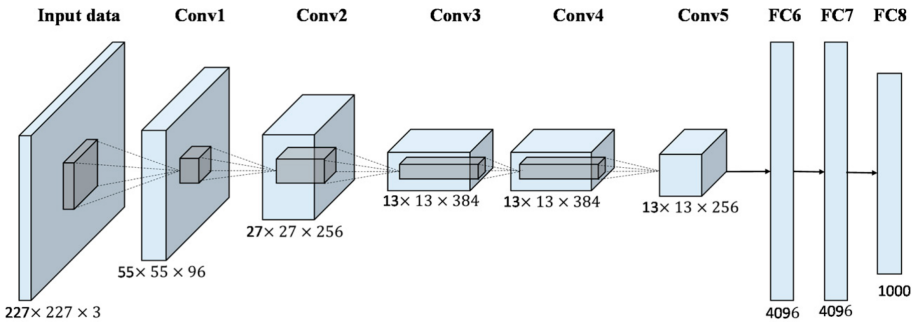


Figure 3-15: The Alex-Net architecture containing 5 convolutional layers and 3 fully connected layers using Relu. Dropout is applied on both fully connected layers. Max pooling layers are added after the convolutional layers.

Alex-Net popularised the use of ReLU activation functions, dropout layers and pooling layers within convolutional neural networks.

Chapter 4

Numerical Experiment: Shack-Hartmann Simulations

This thesis aims to answer the research questions proposed in Chapter 1, in order to develop a new approach to phase reconstruction from raw SH measurements. To answer these questions, two numerical experiments have been performed. The first numerical experiment centers around Shack-Hartmann simulations and sets the stage for the second numerical experiment. In the second numerical experiment, discussed in Chapter 5, we will use these simulated Shack-Hartmann patterns to train/evaluate the proposed neural networks. The outcome of the second numerical experiment will determine the validity of our approach.

In order to create the proposed Deep Learning models, a neural network has a to be trained. Training a neural network requires large amounts of input/output data pairs, such that the model can generalize properly. The purpose of our approach is fast and accurate estimation of a wavefront using its Shack-Hartmann pattern. Thus, our model has to learn the non-linear relationship between Shack-Hartmann patterns (input) and their corresponding combination of Zernike polynomials or phase-map (output). Before we start the training process, which is described in Chapter 5, a fast amount of data pairs has to be generated. This chapter is about data generation, which is vital for training our Deep Learning model. Three different micro-lens array geometries are considered, consisting of 25 (5x5), 256 (16x16) or 900 (30x30) lenses. The 256- and 900 lens micro-lens array's will be subjected to wavefronts originating from the first 32-Zernike polynomials. The 25 lens micro-lens array's will be subjected to wavefronts originating from the first 100-Zernike polynomials. The purpose of the former two data sets is validation of our approach using results achieved in [13] and testing the effect of number of lenses in the micro-lens array. The purpose of the latter data set is testing the accuracy of our approach within a setting where traditional methods fail (i.e. estimation of more Zernike indexes than the available lenses in the MLA).

This chapter is structured as follows: in Section 4-1, a distinction is made between three data sets and the wavefront generation methodology is explained. In Section 4-2, a description is given on the simulation process of the actual Shack-Hartmann patterns and several examples are illustrated. To make the Shack-Hartmann patterns more realistic, each pattern is subjected to noise

and physical parameters are taken into consideration. The type of noise is described in Section 4-3 and the implication of physical parameters is discussed in Section 4-4.

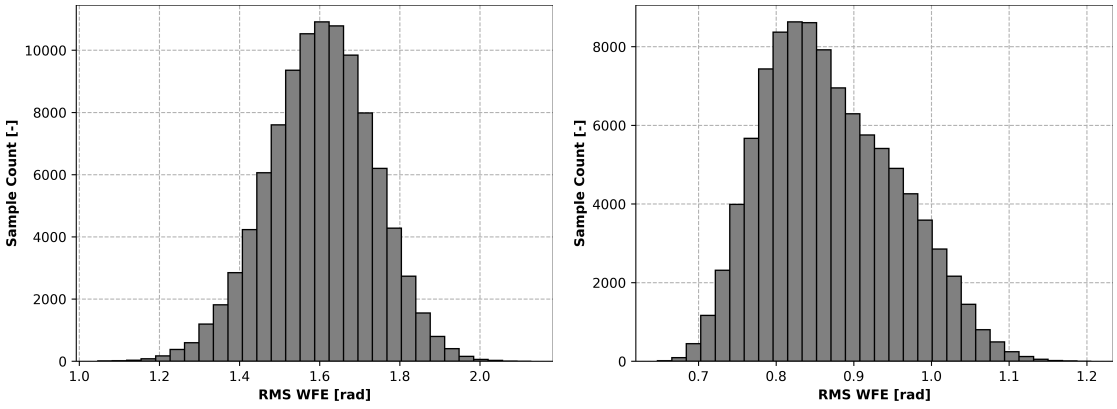
4-1 Data Generation

The Shack-Hartmann patterns are generated in Python using Fourier Principles as imaging model (see Fourier Imaging, Section A-2). The SH-patterns are generated using either a 5x5, 16x16 or 30x30 MLA. Using these three MLA geometries, three different data sets are generated:

- SH-patterns (with 16x16 MLA), from first 32-Zernike.
- SH-patterns (with 30x30 MLA), from first 32-Zernike.
- SH-patterns (with 5x5 MLA), from first 100-Zernike.

For both the 32-Zernike and the 100-Zernike sets, the first coefficient (piston) is set to zero, since this does not influence the PSF. For the first 32-Zernike case, the remaining 31 Zernike coefficients were randomly generated within the range of $[-0.5, 0.5]$ rad for computing each phase map. Combining these indexes yield wavefronts between [1.2:2.0] rad, see Figure 4-1a. The extension of from 32-Zernike to 100-Zernike yields more rapidly varying phase maps due to higher-order abberations.

For the first 100-Zernike, the indexes were generated a bit differently. A random $[-0.5, 0.5]$ range like the 32-Zernike case would yield extremely unstable wavefronts. Hence, the indexes were generated as described in [46]. As a result, the range of wavefronts within this set is [0.6:1.2] rad, see Figure 4-1b. In fact, these values correspond to the wavefront error operation range of the vast majority of imaging system [47]. In general, RMS WFE of less than 0.15 rad can be considered as perfect image quality. Poorer image quality is found with RMS WFE greater than 0.15 rad.

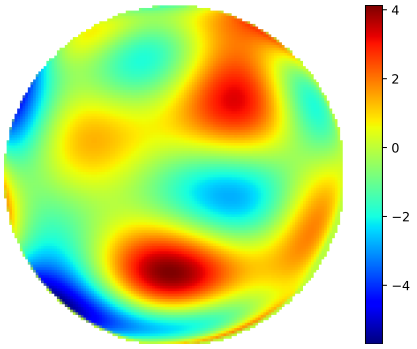
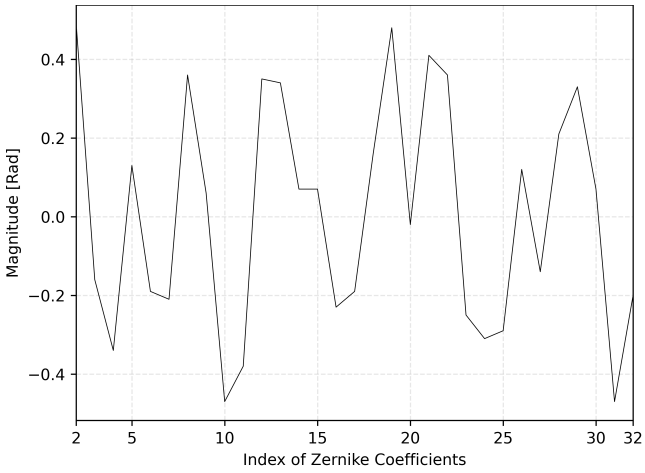
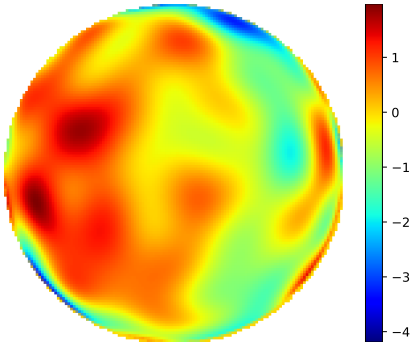
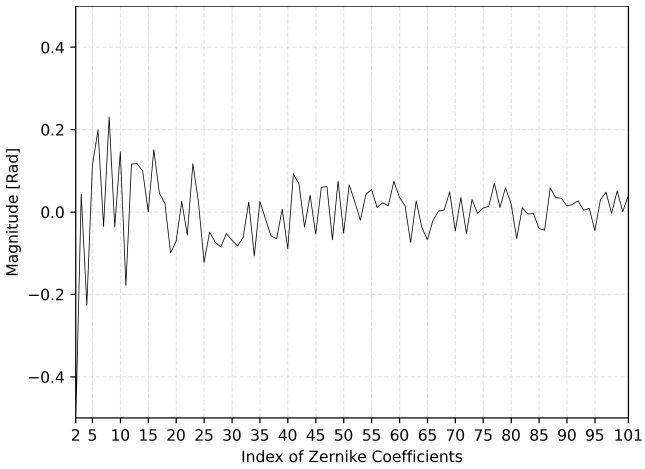


(a) First 32-Zernike data set. **(b)** First 100-Zernike data set.

Figure 4-1: Distribution of RMS WFE across the 32- and 100 Zernike data sets. Both sets roughly follow a Gaussian distribution.

In Table 4-1 two examples are shown denoting phase map and corresponding Zernike indexes.

Table 4-1: Two phase map examples with corresponding Zernike indexes. The top image shows a phase map generated from the first 32-Zernike (piston excluded) with a RMSE of 1.61 *rad*. The bottom image shows a phase map generated from the first 100 Zernike (piston excluded) with a RMSE of 0.91 *rad*. Colorbar is in *rad*.

Phase map	Zernike indexes
	
	

4-2 Shack-Hartmann Patterns

The physical parameters used for simulation are a MLA focal length of 10 mm with a pitch of 150 μm. The observed wavelength has been chosen to be 633 nm and the CCD pixel size has been set to 4.65 μm. The aperture size depends on the amount of lenses used in the MLA. The Hartmann Patterns (HP) used for this project were simulated using the following steps:

- Generate and combine first *n* (32 or 100) Zernike polynomials using random indexes. The Zernike polynomials are generated in the Aotools library [48].
- Sub-divide the phase in the pupil into multiple sub-apertures (s/a) depending on MLA geometry.

- Calculate PSF for every isolated s/a using Direct Fourier Transform.
- Add all PFS's coherently and take squared absolute value of the intensity to obtain the final HP.
- Down sample entire HP to desired neural network input size.

Taking the Discrete Fourier Transform (DFT) of the entire MLA at once, instead of every isolated s/a might be tempting, however such a single-shot DFT poses issues which can be traced back to the definition of the FT (see Section A-2). The complex field at coordinates (u, v) are solely determined by phase and amplitude of the input at spatial frequencies $(f_x = \frac{u}{\lambda z}, f_y = \frac{v}{\lambda z})$. Let us assume an input function h sampled at N points x_n (1D example is only considered since generalization to 2D is trivial). The distance between the points is denoted by Δx. Taking the FT over this function yields a function G, sampled at N points $u = f_x \lambda z$. This gives a step length of

$$\Delta u = \Delta f_x \lambda z = \frac{\lambda z}{N \Delta x}, \tag{4-1}$$

with $\Delta f_x = (N\Delta x)^{-1}$. Suppose we sample the input over $x_{\min} = -N/2 \cdot \Delta x$ to $x_{\max} = (N/2 - 1) \cdot \Delta x$ such that the origin is included. As a result, the FT is given over spatial coordinate u from

$$u_{\min} = -\frac{N\Delta u}{2} = \frac{N\lambda f}{4x_{\min}},$$
$$u_{\max} = u_{\min} + N\Delta u = \frac{(N-2)\lambda f}{4x_{\max}}.$$

Assume now an incoming wavefront which is 10 mm in diameter and we sample it using 8192 steps in each dimension (this is a very high sampling density which will already result in very long computation time). Running the FFT (Fast Fourier Transform) would yield a coordinate plane ranging from -2.5 mm to 2.5 mm, which is still insufficient size since the resulting HP should be about the same size as the original wavefront. Furthermore, such a high sampling density is unnecessary since the sample density of the CCD is fixed. In our simulations, this problem is solved by giving each lenslet its own coordinate system to avoid unpractical sample densities. Each coordinate system is subjected to a FFT using parameters defined in section 4-4.

In the next two pages, Tables 4-2 and Table 4-3 illustrate some random samples taken from the data sets.

Table 4-2: Example phase map and corresponding HP (MLA 16x16 and 30x30). Phase maps are generated by using the first 32 Zernike coefficients. The denoted HP serve as input to the Neural Networks. Colorbars are in *rad*.

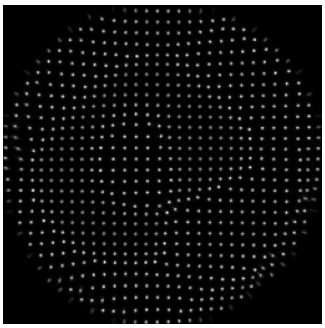
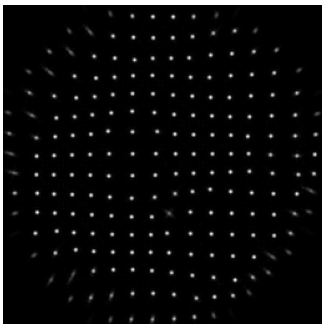
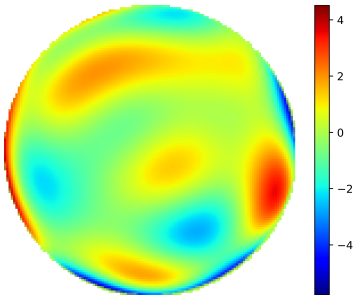
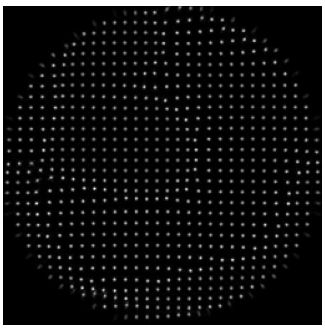
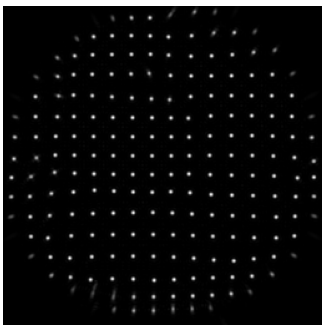
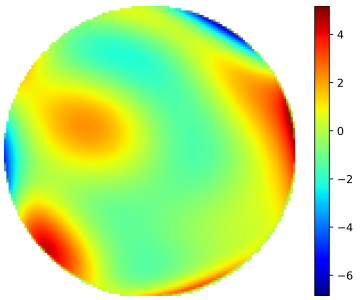
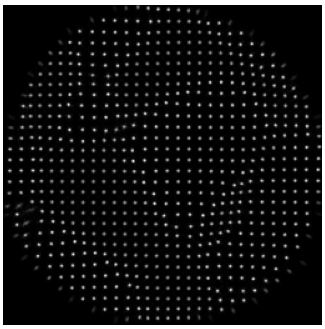
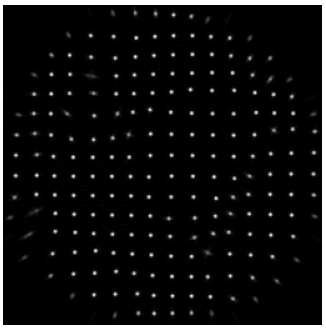
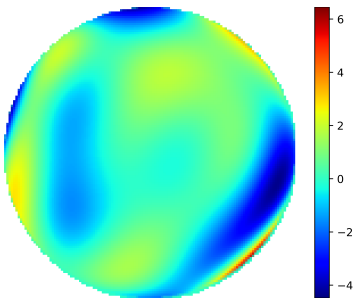
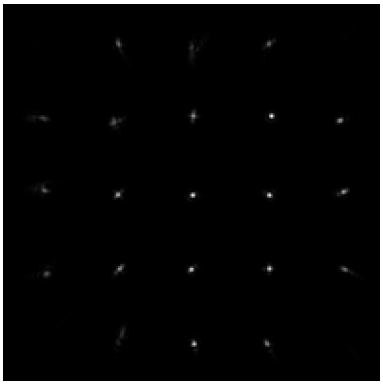
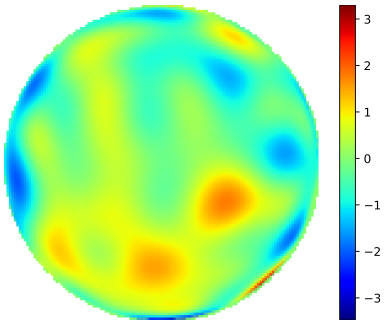
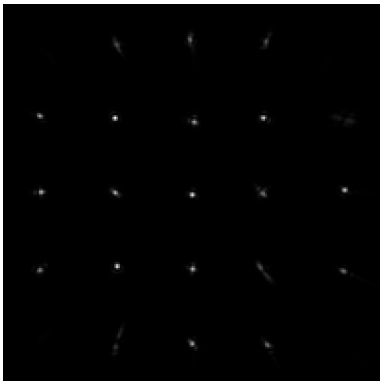
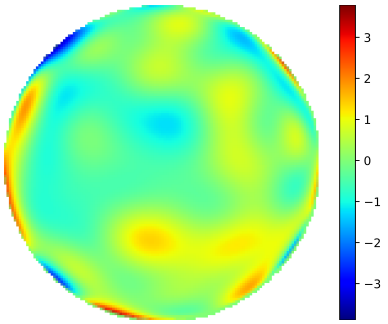
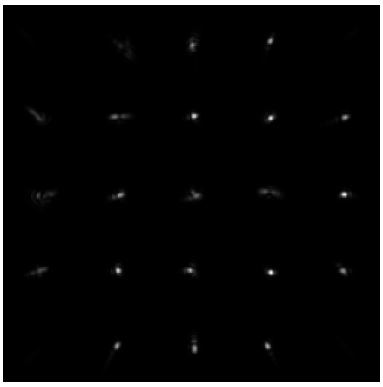
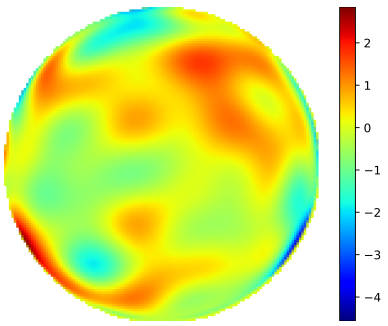
30x30	16x16	Phase map
		
		
		

Table 4-3: Example phase map and corresponding HP (MLA 5x5). Phase maps are generated by using the first 100 Zernike coefficients with statistical model described 4-1. The denoted HP serve as input to the Neural Networks. Colorbars are in *rad*.

5x5	Phase map
	
	
	

4-3 Noise

To make the models more robust to usage in real optical setups, noise has been added to the HP. There are various noise sources that impact the imaging system. In this project the assumption has been made that only the CCD contributes to noise generation. The primary noise contributors [49] are denoted in Table 4-4:

Table 4-4: Quantities used to determine noise levels generated by a CCD.

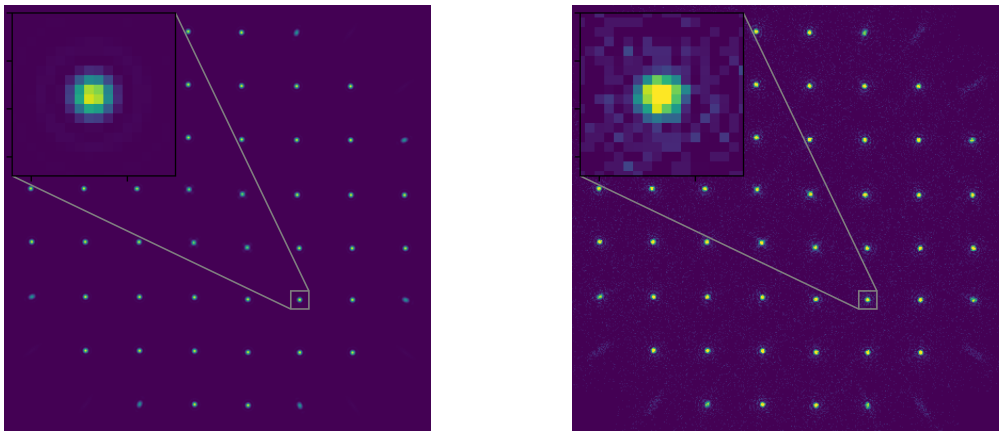
Quantity	Meaning
Q_{e}	Quantum Efficiency
N_{dc}	Dark-current noise
N_r	Read-out noise
N_s	Shot noise
P	Photon flux

Q_{e} refers to the ratio photons hitting the CCD to conversion into electrical impulse. Not all incident photons are actually detected. N_{dc} and N_r refer to the dark-current noise and read-out noise of the CCD (see A-1-8). N_s is caused by inherent statistical variation in the arrival rate of incident photons and is due to the quantum nature of light. Shot noise cannot be reduced by camera design and thus the minimum noise level achievable is always bounded by this noise type. In general, the number of electrons (N) follows a Poisson distribution:

$$P(x) = \frac{N^x e^{-N}}{x!}. \tag{4-2}$$

This is because photons are emitted randomly and independently. In case $N \gg 1$, this distributions skewed towards a Gaussian distribution $\mathcal{N}(N, N)$. Hence, the shot noise scales with $\sqrt{N}$.

For this project, a short exposure time is assumed, causing shot noise to be the dominant factor ($N_s \gg \mathrm{N_{dc}}, \mathrm{N_r}$). This renders $\mathrm{N_{dc}}, \mathrm{N_r}$ negligible. Hence, the simulated HP are dictated by Poisson statistics. On average, a Peak Signal to Noise Ratio (PSNR) of 27 dB is assumed. This is illustrated in Figure 4-2.



(a) HP without noise. **(b)** HP with noise.

Figure 4-2: Each HP is subjected to Poisson noise for more realistic simulations. On average, a PSNR of 27 dB is reached.

4-4 Anti-aliasing

When simulating a SH-pattern, it is important to keep in mind the physical limitations of sensors. Using arbitrary amount of pixels for the SH-pattern will yield in unrealistic results [50]. For a proper SH-pattern simulation without aliasing the pixel size should be small enough such that the Nyquist criterion holds:

$$\frac{D}{f}\frac{2\pi}{\lambda} \leq \frac{\pi}{s}. \tag{4-3}$$

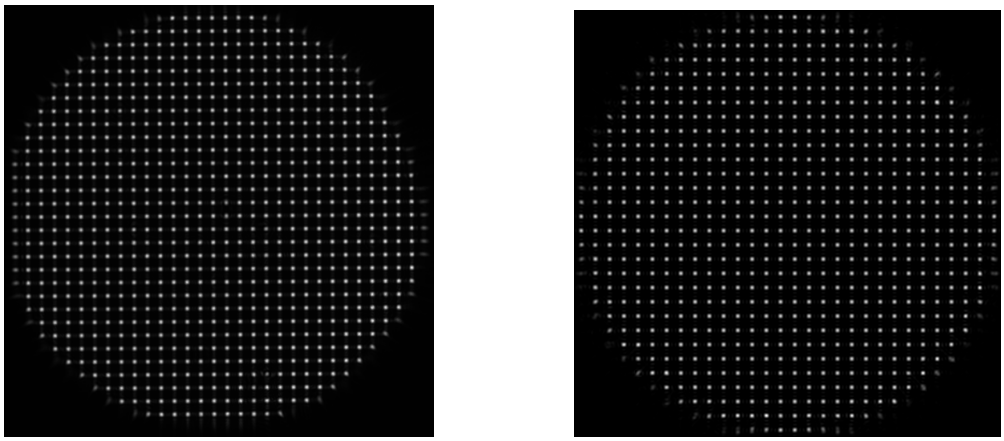
Where s is the pixel size (of CCD), D the total aperture size and f the focal length and λ the light wavelength. Two other important parameters to keep in mind are the pitch and amount of lenses the SH is constructed with. Pitch meaning the distance between the center of two neighbouring lenses inside the micro-lens array (MLA). These two factors determine the aperture size of the MLA.

Consider the following example, given a MLA of 16x16 lenslets with a pitch of $0.15\,mm$, focal length $10\,mm$ and aperture diameter $2.4\,mm$ $(16 \cdot 0.15 \cdot 10^{-3})$. Also given that the CCD has a sampling rate of $4.65\,\mu m$ and the incoming light has a wavelength of $\lambda = 633$ nm. Inserting these parameters into Equation 4-3 bounds total aperture size to a maximum of $\approx\ 1.14mm$ to avoid aliasing. Although the SH-pattern could fit inside a grid of 517 pixels, it should be sampled with about 3 times smaller pixel size to avoid aliasing. Hence, the simulation grid for the pupil field should be about 1551x1551.

4-5 Summary

First we discussed the type of wavefronts used to generated the Shack-Hartmann patterns, using a combination of 32- or 100-Zernike coefficients. Then, a list of steps was provided, taking the reader

through the process of simulating a Shack-Hartmann pattern. The quality of Shack-Hartmann patterns generated through a physical optical setup, is always influenced by noise and hardware limitations. Thus, some physical parameters were introduced to the simulated patterns to make them more realistic. Figure 4-3a shows a Shack-Hartmann pattern generated inside a physical setup, Figure 4-3b shows a Shack-Hartmann pattern simulated in Python.



(a) Real Shack-Hartmann pattern. **(b)** Simulated Shack-Hartmann pattern.

Figure 4-3: Real and Simulated Shack-Hartmann pattern from a flat wavefront. **(a)** has a pitch, focal length and pixel size of 150 μm, 10 mm and 5.2 μm respectively. **(b)** has a pitch, focal length and pixel size of 150 μm, 10 mm and 4.65 μm respectively.

Concluding, three different data sets are simulated which vary in micro-lens array setting and wavefront aberrations. In the next chapter, we will show how these generated data sets are utilized to train different neural networks.

Chapter 5

Numerical Experiment: Training and Evaluation

In this chapter, we will show the training and evaluation of three custom neural network architectures: Xception, Alex-Net and U-Net. These custom neural networks have been adapted to fit our desired output and each are trained with respect to three data sets (as mentioned in Chapter 4). This yields 9 models in total, 3 neural networks times 3 data sets equals 9 models. We can evaluate these models post training by taking the residual error between predicted wavefront and original wavefront. Comparing these residual errors, we can determine the influence of MLA geometry, data set size and network architecture. Each network is trained with respect to a specific loss function, depending on the network type.

In the introduction of this chapter, the three networks and their training schemes are briefly discussed. In Section 5-2, the network results regarding the two '32-Zernike' data sets are reviewed and a single prediction is illustrated. In Section 5-3 the network results regarding the '100-Zernike' data set are discussed and a single prediction is illustrated. Finally, the Xception network trained with respect to the '100-Zernike' data set is used for a more comprehensive analysis. The robustness of this model is evaluated using different noise types and noise intensities. Furthermore, the static and dynamic closed-loop behaviour is modeled.

5-1 Introduction

All data sets are split up into 100.000 training images, 1000 test images and 1000 validation images respectively (see Section A-3-1). The codes were implemented in Python and were executed on a computer with a single Radeon RX Vega GPU. Keras [51] was used as Deep Learning framework within Python. PlaidML was used in combination with Keras to enable Keras to run on a AMD GPU. PlaidML is an advanced and portable tensor compiler. To test the influence of training set size, the models are also trained with respect to two smaller sets sizes using the 16x16 MLA data set.

Some adjustments to the Xception network (see Section 3-4-2) were required such that it was capable of Zernike regression. The top fully connected layer is replaced since on default it outputs 1000 classes, while we need 32 (or 100). A dropout layer is added before the last fully connected layer we just replaced. The dropout layer has a dropout ratio of 0.5 which greatly reduces overfitting of the model. The new fully connected output layer has 32 (or 100) neurons and has a linear activation function. We need a linear activation in the last layer since this enables regression. The input layer has been modified to accept images of size 256x256x1 (length,height,channels). During training, the Xception network will be evaluated using the following loss function:

$$\text{Loss} = \sqrt{\frac{\sum_{i=1}^{N} (c_i - c_i')^2}{N}}, \tag{5-1}$$

where N is the number of predicted Zernike coefficients and c_i and c_i' are the exact- and predicted Zernike coefficients respectively. Evaluation of the network post training will be done using the Root Mean Square Error (RMSE), which in our case, is equal to the loss function but without averaging.

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} (c_i - c_i')^2}, \tag{5-2}$$

Averaging should not be done by nature of the Zernike [52].

The second Neural network, based on Alex-Net (see Section 3-4-3), has also been modified for our goal. The last fully connected layer has been reduced in size to fit our 32- or 100 Zernike indexes and has a linear activation function. Furthermore, two drop-out layers with ratio 0.25 have been added after the first and second fully-connected layers. Alex-net has less convolutional operations than Xception but is still capable of Zernike regression [53]. Fewer convolutional layers yield faster training- and prediction times. Like Xception, Alex-Net is evaluated during training using the loss function defined in Equation 5-1.

The third neural network, U-Net (see Section 3-4-1), enables pixel-wise classification. Unlike Xception and Alex-Net, U-Net does not estimate Zernike indexes, but is trained to directly output a one-to-one pixel-wise estimation of the phase-map. From the estimated phase-map, the Zernike indexes can be recovered by projecting the phase onto the Zernike orthogonal basis

$$c_i = \frac{< \theta(x,y), Z_i(x,y) >}{< Z_i(x,y), Z_i(x,y) >}. \tag{5-3}$$

Although Zernikes polynomials are normalized, $< Z_i(x,y), Z_i(x,y) > \neq 1$ in numerical simulations since they are sampled on a grid.

The input/output layer has been modified to accept/output images of size 256x256x1. During training, the network is evaluated with respect to the root mean squared wavefront error (RMS WFE) between the real- (θ) and predicted phase (θ')

$$\text{Loss} = \sqrt{\frac{1}{N(\rho)} \sum_{i,j \in \rho} \left(\theta\left(x_i, y_i\right) - \theta'\left(x_j, y_j\right)\right)^2}, \tag{5-4}$$

where ρ is the sampled pupil aperture and $N(\rho)$ is the total number of pixels within the apertures diameter. Post training the network is evaluated with respect the same RMSE metric as the previous networks. This is achieved by recovering the Zernike indexes from the estimated phase-map.

Figure 5-1 gives a broad overview of the training process. During training, the goal is to minimize the defined loss function. This loss function is an important performance metric indicating the accuracy of the model. The smaller the loss, the closer the original- and predicted wavefronts are related.

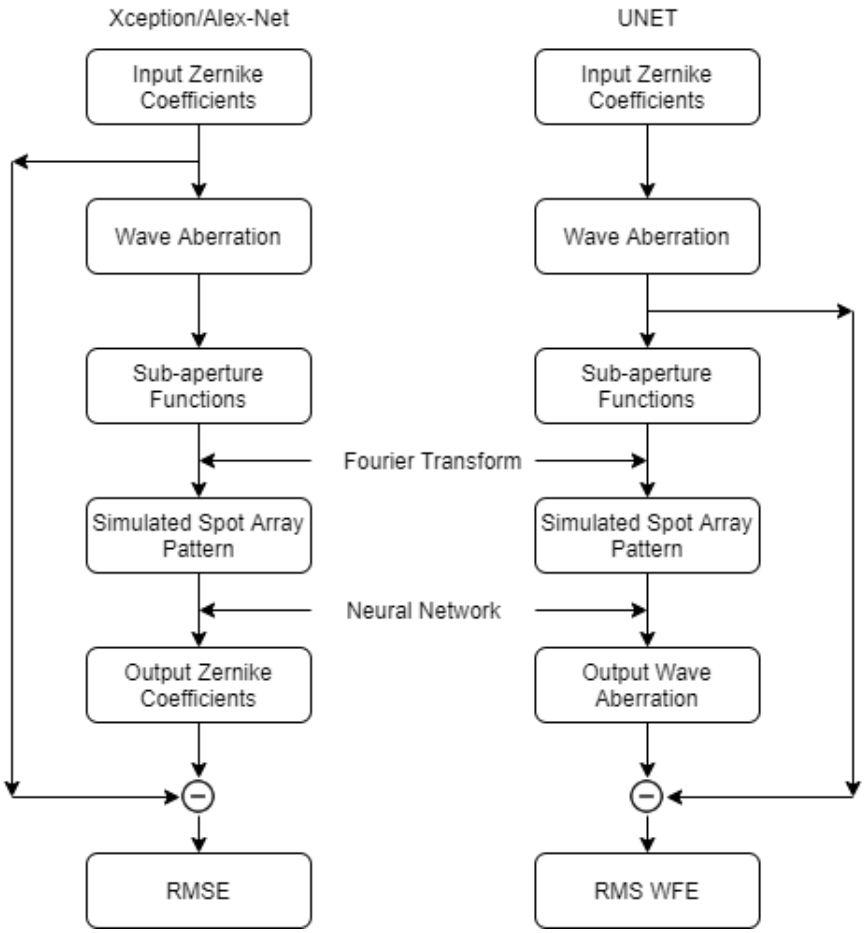


Figure 5-1: Network training flow of operations used for loss calculations. During training, the averaged RMSE or RMS WFE is minimized. First, a random set of Zernike coefficients are generated. Using these indexes, a wavefront is generated and subjected to a micro-lens array, causing sub-aperture functions. Taking the Fourier Transform for each s/a function, results in a Shack-Hartmann pattern. This focal spot pattern is used by the neural network to estimate Zernike coefficients or a phase-map, depending on the network. (**left**) Flowchart indicates the flow of operations for Xception and Alex-Net. (**right**) Flowchart indicates flow of operations for U-Net.

The *ADAM* optimizer was chosen as primary optimization routine for both networks. It has remarkable computational efficiency and handles learning rate optimization automatically, which saves considerable time hyper-tuning this parameter [34]. *ADAM* is initialized to a user-defined value and then updated according to values of the gradient. Total allowed trainable epochs was

50, however, whenever the validation loss did not decrease for 5 epochs, the training sequence was stopped automatically. This was done to prevent unnecessary long training times and overfitting of the model. Table 5-1 shows the influence of network design on training- and inference time.

Table 5-1: Table showing amount of trainable parameters, time spend on a single epoch during training (100.000 samples) and post training prediction time from HP to phase map. Computed using a single Radeon RX Vega GPU.

Network	Parameters	Time per Epoch (h)	Inference (ms)
Xception	20,869,895	1.2	13
U-Net	485,673	0.45	7
Alex-Net	33,952,319	0.15	2

Table 5-1 reveals that the size of a network is not necessarily responsible for longer inference times. It is the amount of multiplications done within the network that induces longer inference- and training times. All of the above networks achieve competitive operation speeds, compared to traditional Shack-Hartmann wavefront sensing methods which may take around 17.6 ms per pattern [54].

5-2 Training Neural Networks on 32-Zernike

5-2-1 Xception

Figure 5-2 denotes the learning curve obtained using the following learning protocols: ADAM optimizer, learning rate of 1e-4 and batch size of 10. Figure 5-2 shows the validation- and training loss (see Section A-3-1). The model should always perform better on training data than unseen data, however, this is not the case when dropout layers are present in the network. Dropout is only applied on the training set, hence the validation loss remains unaffected.

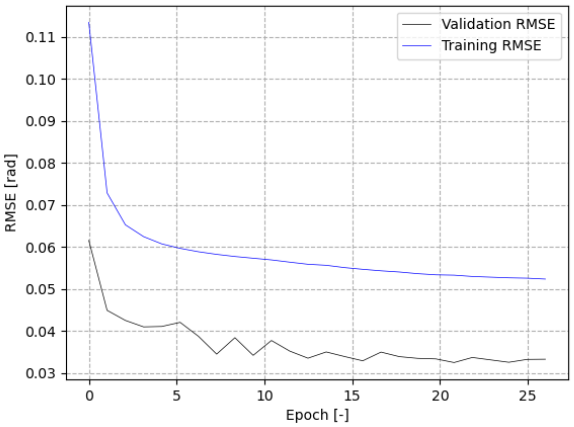


Figure 5-2: Example learning curve showing loss vs. epochs. Obtained from using 100.000 data pairs and 16x16 MLA HP. Validation loss is lower than the training loss due to the dropout layer.

Figure 5-3 shows the effect of training set size on the performance of the resulting model. A larger training set size has a positive effect on the models performance, however the percentage of improvement does decrease with larger data set sizes. Meaning a lot of improvement can be gained going from 10.000 to 50.000 set sizes, but less from 50.000 to 100.000 set sizes. Table 5-2 presents the post training evaluation of the model, showing the RMSE over the test set using two MLA geometries. Increasing the number of lenses from 256 to 900 did not yield significantly better results.

Table 5-2: The RMSE (Equation 5-2) of Xception over the 1000 samples test set.

MLA	RMSE $[rad]$	RMSE $[nm]$
16x16	0.17 $\pm$ 0.07	16 $\pm$ 7.2
30x30	0.16 $\pm$ 0.06	15 $\pm$ 6.1

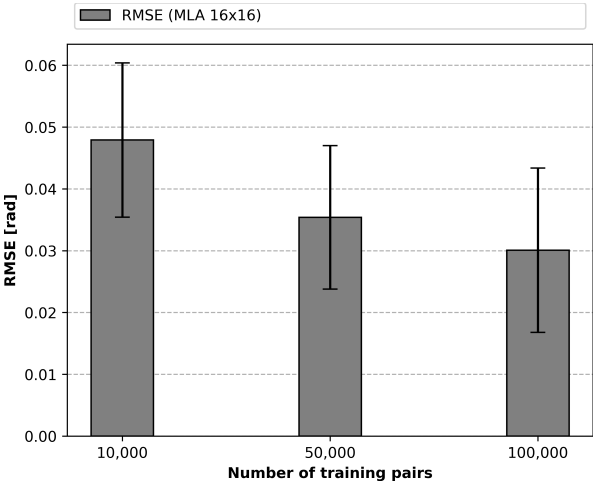


Figure 5-3: Averaged RMSE (using Equation 5-1) vs number of training pairs for three 16x16 MLA models over a 1000 data pair test set.

5-2-2 U-NET

Figure 5-4 denotes the learning curve obtained using the following learning protocols: ADAM optimizer, learning rate of 1e-4 and batch size of 64. Figure 5-4 shows the validation loss (performance model on unseen data) and training loss (performance model on train set).

Figure 5-5 shows the effect of training set size on the performance of the resulting model. A larger training set size has a positive effect on the models performance, however the percentage of improvement does decrease with larger data set sizes. Noticeably, U-Net is less effected by a smaller set size than Xception or AlexNet.

Table 5-3 presents the post training evaluation of the model, showing the RMSE over the test set using two MLA geometries. Increasing the number of lenses from 256 to 900 decreases the error rate by roughly 20 %.

Table 5-3: The RMSE (Equation 5-2) of U-Net over the 1000 samples test set.

MLA	RMSE $[rad]$	RMSE $[nm]$
16x16	0.33 $\pm$ 0.069	33 $\pm$ 6.9
30x30	0.27 $\pm$ 0.056	29 $\pm$ 5.6

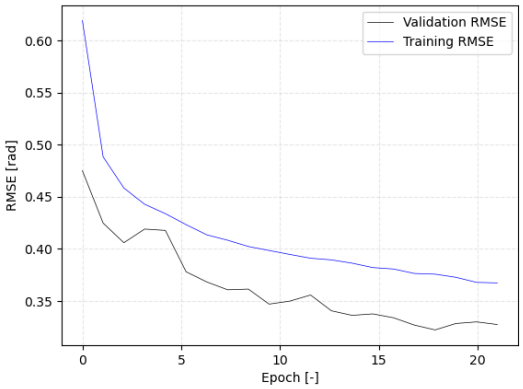


Figure 5-4: Example learning curve showing loss vs. epochs. Obtained from using 100.000 data pairs and 16x16 MLA HP. Validation loss is lower than the training loss due to the dropout layer.

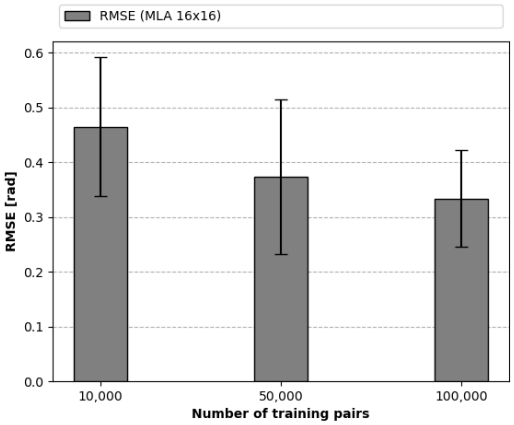


Figure 5-5: Averaged RMSE (using Equation 5-1) vs number of training pairs for three 16x16 MLA models over a 1000 data pair test set.

5-2-3 Alex-Net

Figure 5-6 denotes the learning curve obtained using the following learning protocols: ADAM optimizer, learning rate of 1e-5 and batch size of 128. Figure 5-6 shows the validation loss (performance model on unseen data) and training loss (performance model on train set). As expected, Figure 5-7 shows a decrease in average error when the set size increases.

Table 5-4 presents the post training evaluation of the model, showing the RMSE over the test set using two MLA geometries. Increasing the number of lenses from 256 to 900 does not yield significant reduction in error rates.

Table 5-4: The RMSE (Equation 5-2) of Alex-Net over the 1000 samples test set.

MLA	RMSE $[rad]$	RMSE $[nm]$
16x16	0.38 $\pm$ 0.089	39 $\pm$ 8.9
30x30	0.36 $\pm$ 0.066	36 $\pm$ 6.7

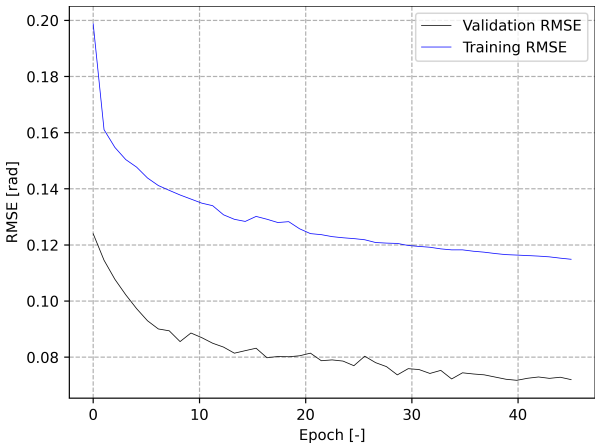


Figure 5-6: Example learning curve showing loss vs. epochs. Obtained from using 100.000 data pairs and 16x16 MLA HP. Validation loss is lower than the training loss due to the dropout layer.

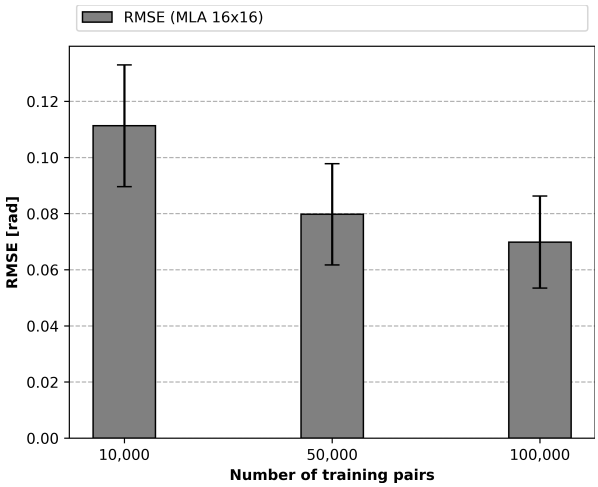
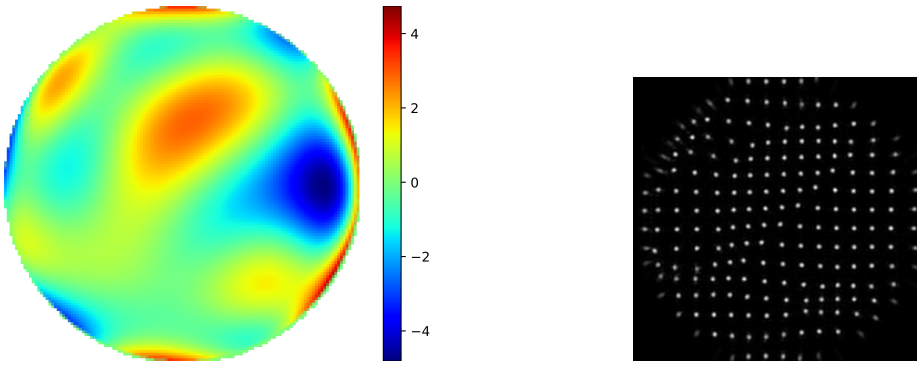


Figure 5-7: Averaged RMSE (using Equation 5-1) vs number of training pairs for three 16x16 MLA models over a 1000 data pair test set.

5-2-4 Prediction Example Networks

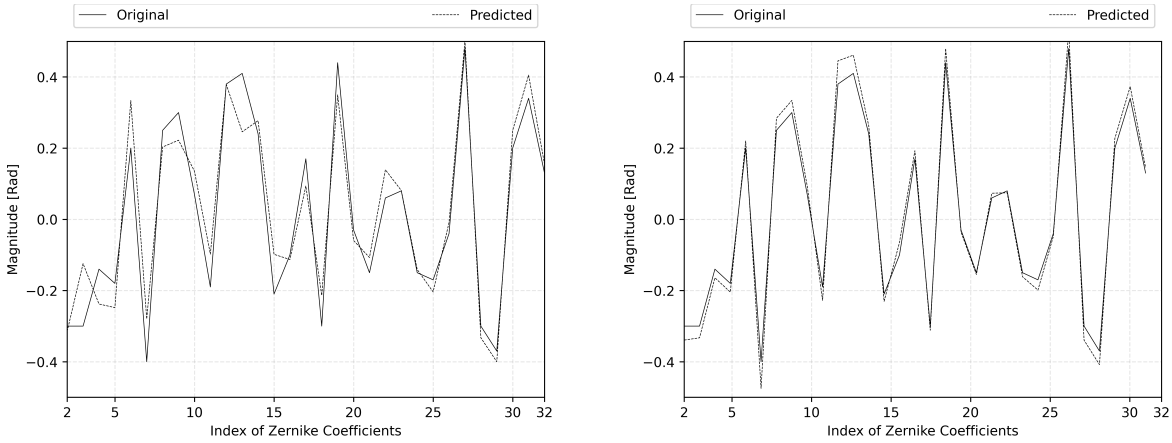
In this section a randomly sampled prediction will be illustrated and discussed. An unseen 16x16 MLA HP (see Figure 5-8b, noise has not been applied yet) is used as input to all three networks. This HP is constructed using a randomly generated phase-map shown in Figure 5-8a. Note that all colorbars are in rad.



(a) **(b)**

Figure 5-8: **(a)** Randomly generated phase-map (θ) using first 32 Zernike indexes, RMSE: 1.5 rad. **(b)** Corresponding noise-less Shack-Hartmann pattern. Noise is added before the image is submitted to the network.

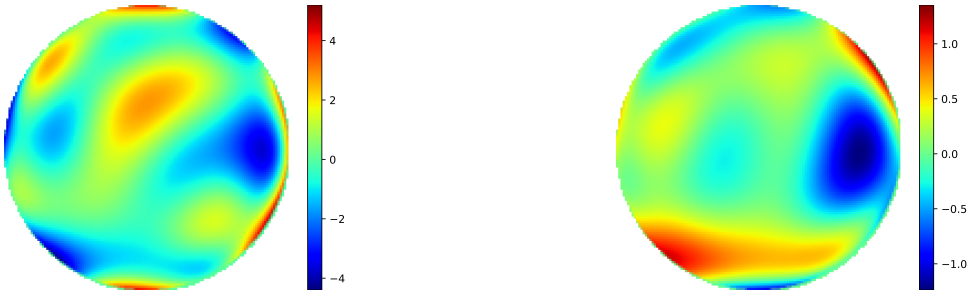
Xception and Alex-Net both estimated Zernike indexes (c_i') which are compared to the real Zernike indexes (c_i) in Figure 5-9.



(a) **(b)**

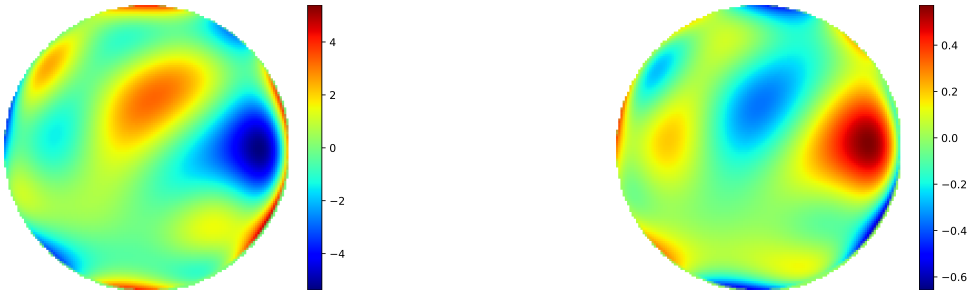
Figure 5-9: Predicted- (c_i') and original (c_i) Zernike indexes from **(a)** Alex-Net, residual RMSE 0.47 rad and **(b)** Xception, RMSE 0.2 rad.

These estimated Zernike indexes can be used to infer an estimated phase-map (θ'), see Figures 5-10, 5-11 . U-NET directly outputs an estimated phase-map, see Figure 5-12.



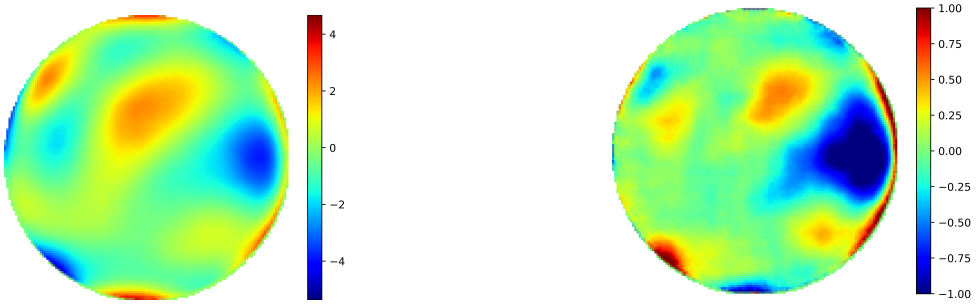
(a) **(b)**

Figure 5-10: **(a)** Predicted Wavefront (θ') Alex-Net. **(b)** Residual wavefront ($\theta - \theta'$). RMS WFE 0.47 *rad*.



(a) **(b)**

Figure 5-11: **(a)** Predicted Wavefront (θ') Xception. **(b)** Residual wavefront ($\theta - \theta'$). RMS WFE: 0.2 *rad*.



(a) **(b)**

Figure 5-12: **(a)** Predicted Wavefront (θ') U-Net. **(b)** Residual wavefront ($\theta - \theta'$). RMS WFE: 0.37 *rad*.

First we notice that especially Figures 5-11b and 5-12b have a similar error profiles, repeating that of the input phase. Figure 5-10b has some components of the input phase, but due to less exact Zernike coefficient estimation shows less commonality. U-Net's residual wavefront shows pixel-to-pixel discontinuities and non-smoothness. However, these pixel-to-pixel variations are small compared to pixel values of the predicted wavefront. Hence, this non-smooth behaviour is negligible.

5-3 Training Neural Networks on 100-Zernike

5-3-1 Xception

Figure 5-13 denotes the learning curve obtained using the following learning protocols: ADAM optimizer, learning rate of 1e-5 and batch size of 10. It shows the validation loss (performance model on unseen data) and training loss (performance model on train set). Figure 5-14 shows a random HP and its resulting predicted Zernike indexes from the Xception model.

Table 5-5 presents the post training evaluation of the model, showing the RMSE over the test set using the 5x5 MLA geometry.

Table 5-5: The RMSE (Equation 5-2) of Xception over the 1000 samples test set.

MLA	RMSE $[rad]$	RMSE $[nm]$
5x5	0.2 ± 0.13	20 ± 13

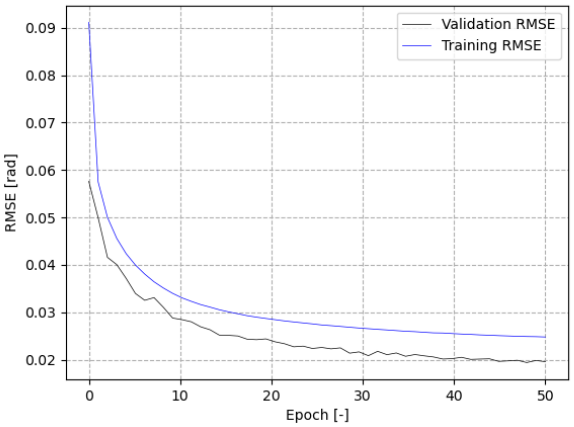
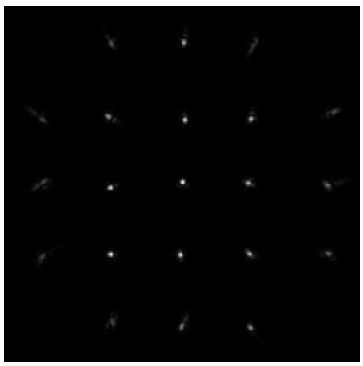
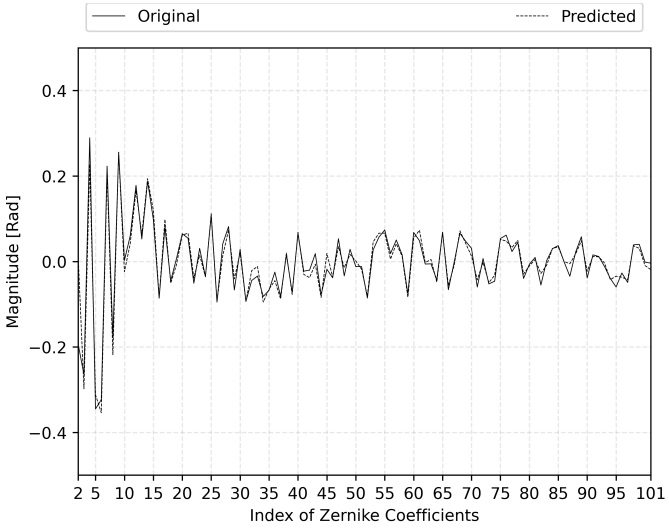


Figure 5-13: Example learning curve showing loss vs. epochs. Obtained from using 100.000 data pairs and 5x5 MLA HP. Validation loss is lower than the training loss due to the dropout layer.



(a)



(b)

Figure 5-14: **(a)** Noise-less Shack-Hartmann pattern (input) and **(b)** predicted- (c_i') and original (c_i) Zernike indexes (output) from the Xception network. Noise is added before the image is submitted to the network. This example has a residual RMSE of 0.15 rad.

5-3-2 U-NET

Figure 5-15 denotes the learning curve obtained using the following learning protocols: ADAM optimizer, learning rate of 1e-4 and batch size of 64. It shows the validation loss (performance model on unseen data) and training loss (performance model on train set). Figure 5-16 shows the predicted phase-map (θ') and the predicted Zernike indexes (c_i') from the predicted phase-map.

Table 5-6 presents the post training evaluation of the model, showing the RMSE over the test set using the 5x5 MLA geometry. Noticeably, the RMSE values between U-Net and Xception are much closer together using the 5x5 MLA than the 16x16/30x30 MLA values.

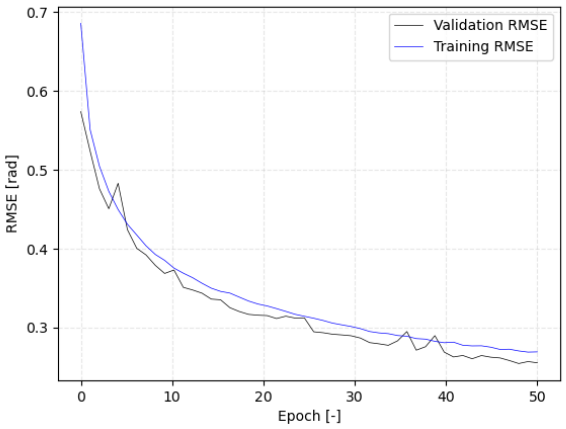
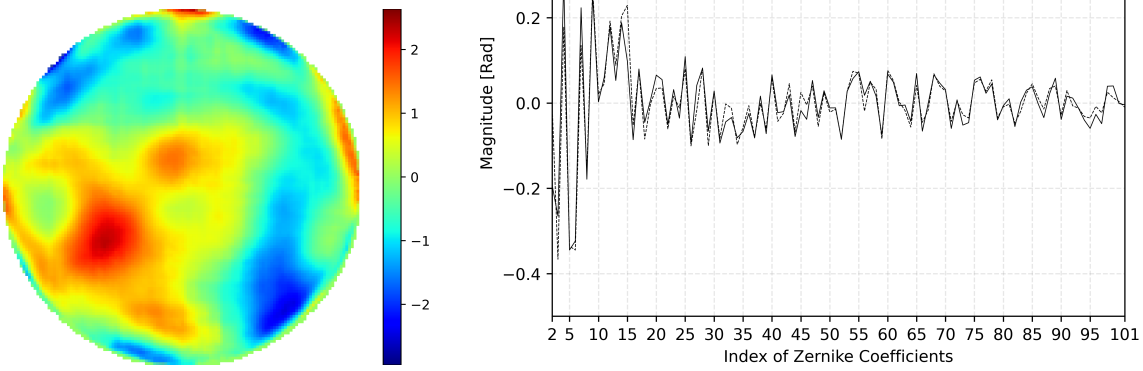


Figure 5-15: Example learning curve showing loss vs. epochs. Obtained from UNET using 100.000 data pairs and a 5x5 MLA HP. Validation loss is lower than the training loss due to the dropout layer.

Table 5-6: The RMSE (Equation 5-2) of U-Net over the 1000 samples test set.

MLA	RMSE $[rad]$	RMSE $[nm]$
5x5	0.26 $\pm$ 0.038	26 $\pm$ 3.8



(a) **(b)**

Figure 5-16: **(a)** Predicted phase-map (θ') from the identical input illustrated in Figure 5-14a and **(b)** predicted- (c_i') and original (c_i) Zernike indexes determined through the predicted phase-map. This example has a residual RMSE of 0.34 rad.

5-3-3 Alex-Net

Figure 5-17 denotes the learning curve obtained using the following learning protocols: ADAM optimizer, learning rate of 1e-4 and batch size of 128. It shows the validation loss (performance model on unseen data) and training loss (performance model on train set). Figure 5-18 shows a random HP and its resulting predicted Zernike indexes from the Alex-Net model. The index plot clearly shows an inability to estimate a larger portion of the Zernike indexes. This is due an insufficient number of convolutional layers within Alex-Net, for this particular input. Table 5-7 presents the post training evaluation of the model, showing the RMSE over the test set using the 5x5 MLA geometry.

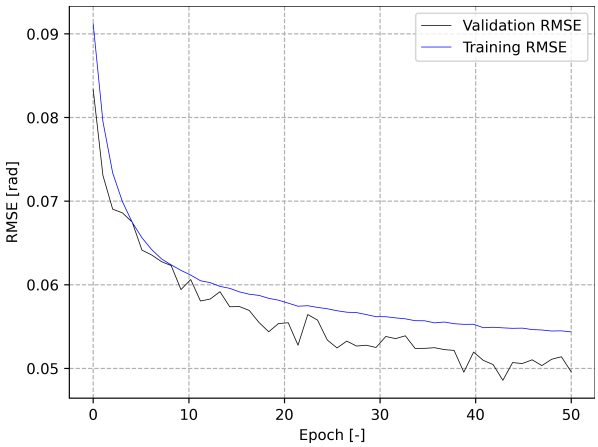
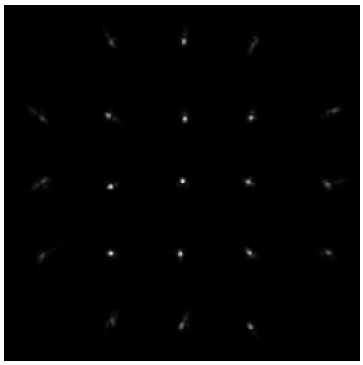


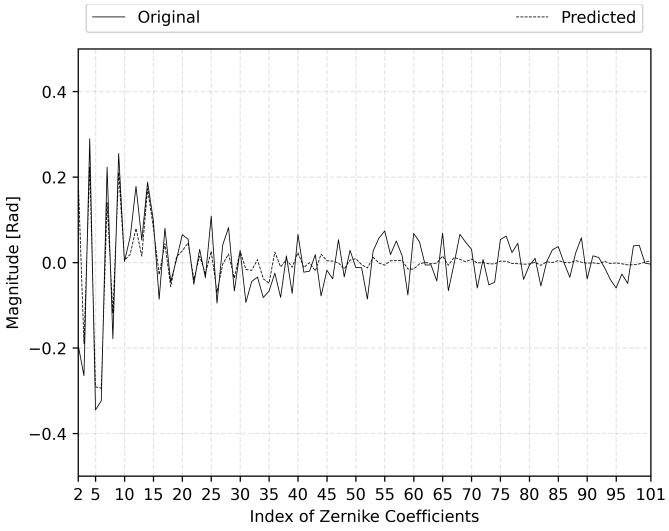
Figure 5-17: Example learning curve showing loss vs. epochs. Obtained from Alex-Net using 100.000 data pairs and a 5x5 MLA HP.

Table 5-7: The RMSE (Equation 5-2) of Alex-Net over the 1000 samples test set.

MLA	RMSE $[rad]$	RMSE $[nm]$
5x5	0.46 $\pm$ 0.043	46 $\pm$ 4.3



(a)

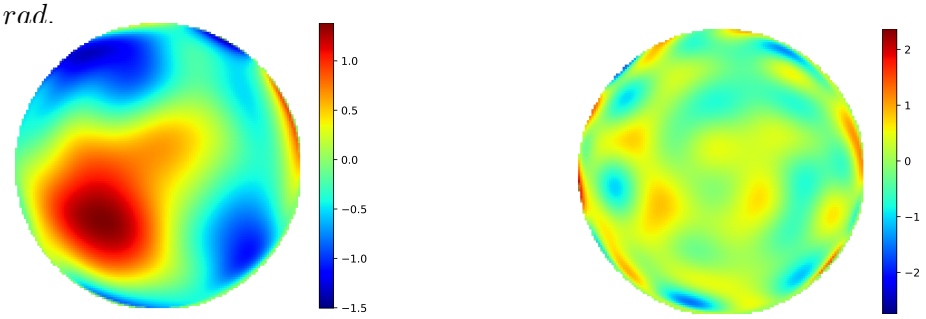


(b)

Figure 5-18: **(a)** Noise-less Shack-Hartmann pattern (input) and **(b)** predicted- (c_i') and original (c_i) Zernike indexes (output) from the Alex-Net. Noise is added before the image is submitted to the network. This example has a RMSE of 0.59 rad.

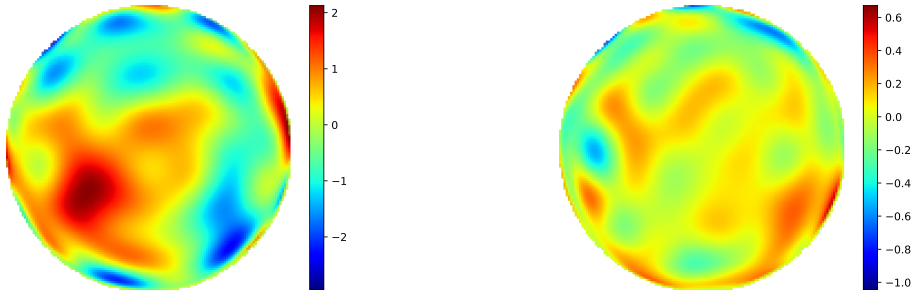
5-3-4 Prediction Example Networks

Figures 5-19, 5-20 and 5-21 show the estimated- and residual wavefronts obtained using the different networks through identical inputs denoted in Figures 5-18a,5-16a and 5-14a. Note that all colorbars are in rad.



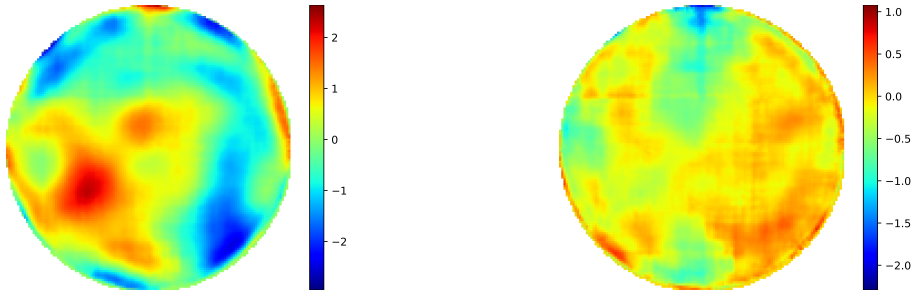
(a) **(b)**

Figure 5-19: **(a)** Predicted Wavefront (θ') Alex-Net. **(b)** Residual wavefront ($\theta - \theta'$). RMS WFE: 0.59 rad.



(a) **(b)**

Figure 5-20: **(a)** Predicted Wavefront (θ') Xception. **(b)** Residual wavefront ($\theta - \theta'$). RMS WFE: 0.15 rad.



(a) **(b)**

Figure 5-21: **(a)** Predicted Wavefront (θ') UNET. **(b)** Residual wavefront ($\theta - \theta'$). RMS WFE: 0.34 rad.

First we notice the high residual error from Alex-Net in Figure 5-19b. This high residual error is the result of poor estimation of the Zernike indexes, see Figure 5-18b. Due to the lack of convolutional filters within Alex-Net, it is incapable properly generalizing. The optimization routine logically prioritized searching for low-order aberrations since these are larger in magnitude. Thus, incorrectly estimated lower-order aberrations would give rise to larger residual errors of the Zernike coefficients.

As a result, the higher-order Zernike polynomials are ignored at the wavefront reconstruction and larger residual errors arise. Secondly, we notice the non-smoothness of U-Net's prediction and residual in Figure 5-21. Although the non-smoothness is more clear in the prediction this time, it does not yield significant residual errors.

5-3-5 Noise Robustness

To evaluate the robustness of the best achieving model (Xception), it will be tested for inputs subjected to a combination of different noise types: Poisson noise, Gaussian noise and ADC (analog-to-digital) noise. ADC noise arises due to the CCD inability to register any continuous spectrum of energy. Values are rounded between 4096 discrete gray levels (12 bits). The model response to different noise levels is denoted in Figure 5-22. Table 5-8 illustrates four key points in Figure 5-22. Testing for these parameters quantifies the applicability of the model in practical setups.

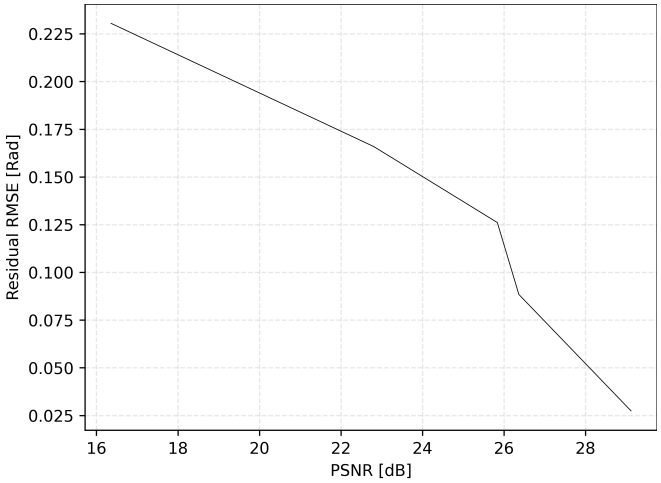
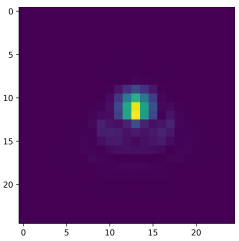
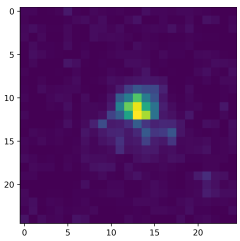
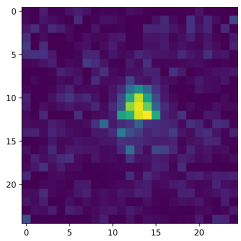
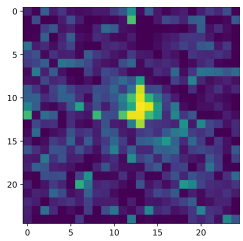


Figure 5-22: The test set is evaluated at different PSNR levels. As the noise intensity increases, the models accuracy decreases.

Figure 5-22 reveals an extreme sensitivity to noise types not included in the train set. Since training conditions assumed short exposure times (i.e Poisson noise), it did not adapt for Gaussian noise leading from longer exposure times. Since the model is not trained to deal with Gaussian noise, it has a great impact on the average accuracy. It is evident that including different noise types into the training procedure is paramount.

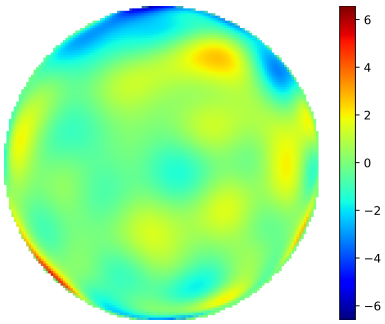
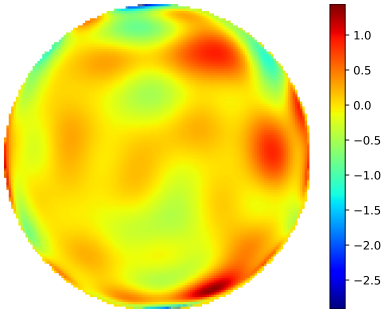
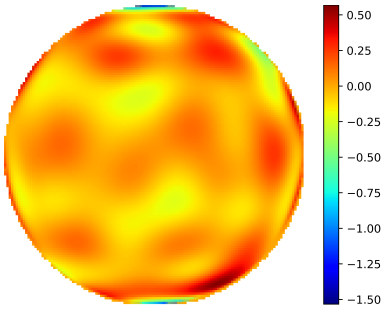
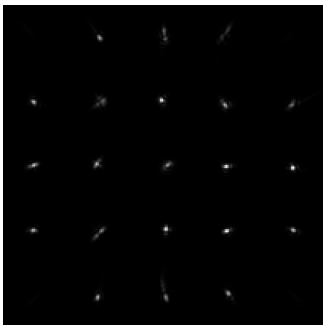
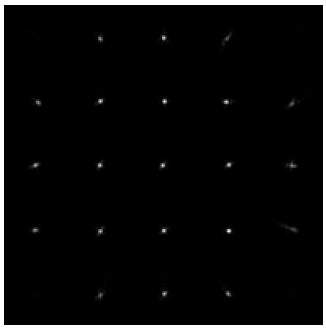
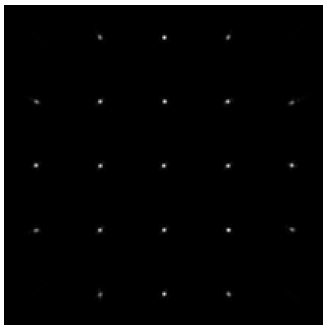
Table 5-8: Illustrations of different PSNR noise intensities taken from a single region of interest located near the center of the MLA.

No noise	29 dB	22 dB	16 dB
			

5-3-6 Closed-Loop Control

To evaluate the closed-loop behaviour of the best achieving model (Xception), multiple iterations are performed using static and dynamic aberrations. Static meaning the incoming wavefront remains unchanged and is only affected by noise. Illustrated in Table 5-9, are three closed-loop iterations using a static aberration. Figure 5-23 denotes the closed-loop behaviour regarding dynamic aberrations.

Table 5-9: Illustrations of three static closed-loop corrections. As the iterations progress, the Hartmann Pattern returns to its diffraction limited form. The RMS WFE's are 1.09 *rad*, 0.31 *rad* and 0.09 *rad* respectively. A vary rapid convergence to a near diffraction limited HP is observed. Colorbars are in *rad*.

Original Wave	Residual Correction One	Residual Correction Two
		
Original HP	**HP Correction One**	**HP Correction Two**
		

Dynamic aberrations, indicate a changing wavefront at every time step due to atmospheric turbulence for example. Since the speed of correction is not instantaneous, a temporal error is introduced to the residual error. This temporal error is due to time lag, introduced by the readout of the wavefront sensor and the inference time of the model. In the example illustrated in Figure 5-23, the incoming wavefront is subjected to moving Kolmogorov turbulence [48] (see Section A-1-7) at each time step.

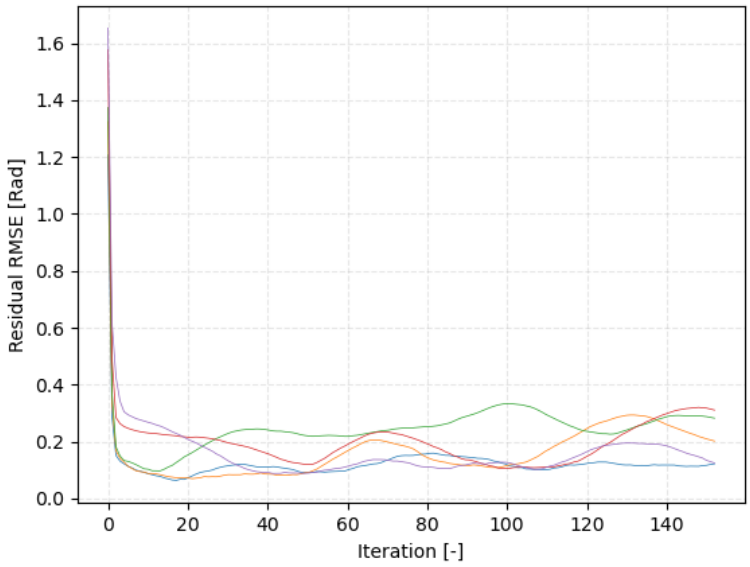


Figure 5-23: Closed-loop performance of the (5x5 MLA) Xception model to dynamic aberrations. The ratio $\frac{D}{r_0}$ (Frieds parameter r0 and aperture size D) equals 6, with a wind speed of 0.5 $\frac{m}{iteration}$. Each line is a separate test, using a different randomized phase screen and reveal stable closed-loop behaviour.

Figure 5-23 indicates stable closed-loop behaviour for the 25 sub-aperture MLA Xception model. It should be stated that traditional Shack-Hartmann wavefront reconstruction algorithms would not be able to stabilize such a feed-back system. These type of algorithms are unable to reliably reconstruct 100-Zernike from a 25 sub-aperture MLA Adaptive Optics system [2].

5-4 Summary

In this chapter we discussed the second and last numerical experiment. In Section 5-2 we showed the training process of six Deep Learning models and evaluated them on their ability to reconstruct 32-Zernike. These six DL models emerged from different MLA geometries and different NN architectures. The Xception network trained on 30x30 MLA HP and 100.000 data pairs achieved the highest accuracy. Furthermore, we noticed that mainly U-Net gained significant accuracy by increasing the amount of lenses in the MLA (from 256 to 900).

In Section 5-3 we showed the training process of three Deep Learning models and evaluated them on their ability to reconstruct 100-Zernike. Just like the 32-Zernike case, the Xception network achieved the highest accuracy. Alex-Net experienced difficulties estimating 100-Zernike from a 5x5 MLA. More thorough analysis of the Xception network revealed a particular high sensitivity to unknown noise types. Furthermore, closed-loop simulations were performed using static and dynamic aberrations. Both cases showed stable closed-loop performance.

Chapter 6

Conclusion

In this Master Thesis, a new approach to phase reconstruction from the SH measurement is proposed. We have shown that Deep Learning techniques in combination with a micro-lens array can surpass traditional SH phase reconstruction methods and alleviate their current limitations. The proposed method uses the entire Shack-Hartmann Pattern as input to a Neural Network, making the centroiding algorithm obsolete. Furthermore, it is shown that Deep Learning models allow for better estimations of the phase-map compared to traditional methods, since they are less depended on MLA size.

Multiple scenarios have been considered, using different MLA geometries (5x5,16x16 and 30x30) with different neural network architectures (Alex-Net, Xception and U-Net). For all cases, a phase-map is constructed by projecting the phase onto an orthogonal basis (Zernike basis). This phase-map is used in combination with three MLA geometries to create three data sets of Hartmann Patterns. Alex-Net and Xception have been trained to estimate Zernike coefficients from such a HP. U-Net on the other hand, is trained to directly output a pixel-wise estimation of the phase-map.

The 16x16 and 30x30 MLA data sets are constructed using phase-maps resulting from the first 32-Zernike. Six models emerge from these MLA/aberrations settings. All networks perform exceptionally well, although no significant performance gain is observed going from a 16x16 MLA to a 30x30 MLA. *We find that the MLA as preconditioner serves as a viable alternative to the proposed preconditioners mentioned in [13].* The 5x5 MLA data set encompasses the first 100-Zernike and as a result, three models are obtained. Alex-Net proofed unable to reconstruct 100-Zernike properly. *We find that in all of the tested cases, the Xception custom network outperforms the other networks in terms of accuracy. Alex-Net has the fastest operation speed.* Further investigation on the trained Xception network shows stable closed-loop performance. Traditional algorithms would only reliably be able to reconstruct 25-Zernikes from such a MLA geometry. *We find that the tested CNN's alleviate this limitation, showing near perfect reconstruction capabilities.*

6-1 Future work

The natural next step would be implement the models in a real physical optical setup, testing their accuracy not only in simulation. Such tests were beyond the scope of this project, but are paramount in testing the methods validity. As the noise robustness analysis showed, certain effects left out in simulation, by accident or by choice, could significantly impact the phase reconstruction. A solution to this would be to gather training data within an optical setup, automatically including physical effects inside the model. A drawback of this approach is that the model would generalize for that setup specifically and could not be share amongst peers. However, techniques like transfer learning could somewhat alleviate this problem.

A common issue in AO systems involving SH phase reconstruction is the alignment and calibration of the sensor which often contribute to systematic errors. Parameters like MLA orientation or distance between lenslet and detector can impact the accuracy significantly. Orientation calibration can be done by recording reference centroids using an ideal light source. Finding the correct distance between MLA and detector can be done by comparing known wavefronts against measurements made by the WFS. It is unavoidable to completely remove all of these effects, hence they should be included to some degree in the Deep Learning model. One approach to include some of the effects is through data augmentation, where each input image is copied several times and subjected to different effects. This does yield a significantly larger data set and training time.

This thesis proofed that Deep Learning models are less reliant on the amount of lenses in the micro-lens array. However, due to time constraints we were unable to perform a comprehensive statistical analysis of the relationship between lenses and Zernike index estimation. Further investigation on this relationship would yield a more sophisticated and applicable method.

Appendix A

Definitions

A-1 Optics

A-1-1 Point Source

A point source is an infinitely small emitter of light which radiates uniformly in all directions.

A-1-2 Optical Path Length

The Optical Path (OPL) is the integral of the refractive index of the medium along the path traveled by light. Which is analogous to traveled time.

$$\mathrm{OPL} = \int_P n(r)\mathrm{d}r \tag{A-1}$$

A-1-3 Wavefront

Wavefront is a surface of a constant OPL from the source [24]. A wavefront is perpendicular to the rays from a point source. A point source at infinity generates a collimated (flat) wavefront.

A-1-4 Abberations

Optical abberations are deviations from a diffraction limited (ideal) wavefront and cause degradation of image quality. Abberations are caused by optical components due to manufacturing errors or the simplified shape of the component. Misalignment, thermal interaction or atmospheric turbulence also cause abberations.

A-1-5 Point Spread Function

The Point Spread Function (PSF) describes the response of an optical system (with circular aperture) to a point source. It is the 2D irradiance profile in the image plane of a point source object. It is a very powerfull mathematical concept in the field of Fourier Optics and other imaging fields. The degree of spreading of the imaged point is a measure of quality of the imaging system and is defined as:

$$\mathrm{PSF}(x,y) = |\mathcal{F}(w(x,y))|^2 \tag{A-2}$$

and,

$$w(x,y) = A_p(x,y)e^{j\varphi(x,y)} \tag{A-3}$$

where $\mathcal{F}$ is the Fourier transform, A_p the transmittance function of the pupil and $\varphi(x,y)$ the wavefront phase over the entrance pupil.

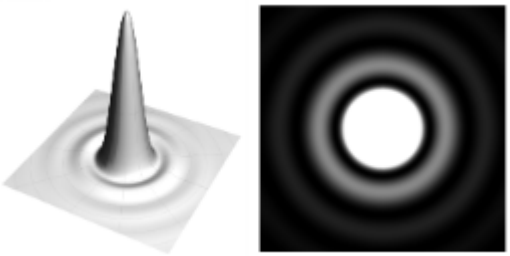


Figure A-1: (left) 3D irradiance profile of a PSF. (right) overexposed 2D intensity distribution. [24]

When a point source is imaged inside a perfect optical system (i.e incoming wavefront phase not subjected to any optical aberrations), the image is shaped by the diffraction and the resulting PSF is called the Airy Disk [24].

$$p_0(\boldsymbol{\alpha}) = \frac{\pi D^2}{4\lambda^2}\left(\frac{2J_1(\pi D|\boldsymbol{\alpha}|/\lambda)}{\pi D|\boldsymbol{\alpha}|/\lambda}\right)^2 \tag{A-4}$$

where the quantities are defined in Table A-1.

Table A-1: Quantities used in Equation A-4

Quantity	Meaning
α	angular coordinate
$p_0(\alpha)$	light intensity in the focal plane
λ	wavelength of light
D	diameter of the telescope aperture
I_1	Bessel function of the first kind

The first minimum from the center (the first dark ring) as an angular distance of $1.22\lambda/D$ from its center. This describes a fundamental limit to the resolution power of a optical instrument. This limit is known as the Rayleigh resolution criterion and states that two objects should be separated at least half the diameter (D) of the airy disk to be resolvable.

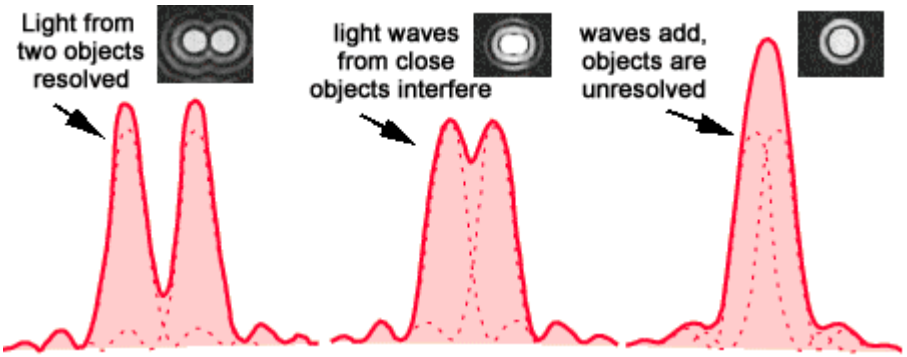


Figure A-2: Schematic of a diffraction limited system: as the distance between the two airy disks decrease, it becomes increasingly difficult to distinguish the two points.[2]

The diffraction-limited point spread function can be used as optimisation quality metric for image-based AO systems. This metric is called the Strehl-ratio and is the ratio of the maximum intensity of the actual point spread function to a perfect point spread function.

A-1-6 Zernike Polynomials

Zernike Polynomials are a set of polynomials which are orthogonal to the unit disk. A weighted sum of these functions can describe the shape of a waveform. Depending on the direction of the wave, Zernike polynomials can be odd or even and are defined using polar coordinates ρ, θ (or Cartesian) as:

$$Z_n^{\pm m}(\rho,\theta) = R_n^m \begin{cases} \cos(m\theta) & l = \pm m > 0 \\ \sin(m\theta) & l - \pm m < 0 \\ 1 & m = 0 \end{cases} \tag{A-5}$$

$$m, n \in \mathbb{Z}_{\geq 0}, \quad n \geq m, \quad n - m \in 2\mathbb{Z} \tag{A-6}$$

Hence, the product of the radial Zernike polynomial $R_n^m(\rho)$ with the azimuthal harmonic component of frequency *m* yields the Zernike polynomial.

The radial polynomials are given by:

$$R_n^m(\rho) = \sum_{s=0}^{\frac{n-m}{2}} \frac{(-1)^s (n-s)!}{s!\,(\frac{n+m}{2}-s)!\,(\frac{n-m}{2}-s)!} \rho^{n-2s} \tag{A-7}$$

The numbering of Zernike polynomials is done through the indexes *n* and *m*, radial- and azimuthal order respectively. Noll [19] proposed a different notation, using a different normalization and index numbering J (or mode).

$$J = \frac{n(n+2)m}{2} \tag{A-8}$$

Ordering the polynomials first by index *n* and the absolute value *m*. The first Zernike modes form basic optical shapes like, Piston (Z_0^0), Tip (Z_1^{-1}), Tilt (Z_1^1) or Defocus (Z_2^0). Figure A-3 denotes some of these lower-order modes (in Cartesian coordinates) and their effect on the PSF formation.

Index	Noll's ordering	Name	Expression	Shape	PSF
$Z_0^0(x,y)$	1	piston	1		
$Z_1^1(x,y)$	2	tip	x		
$Z_1^{-1}(x,y)$	3	tilt	y		
$Z_2^0(x,y)$	4	defocus	$2x^2+2y^2-1$		
$Z_2^2(x,y)$	5	astigmatism	x^2-y^2		
$Z_2^{-2}(x,y)$	6	astigmatism	$2xy$		
$Z_3^1(x,y)$	7	coma	$3x^3+3y^2x-2x$		
$Z_3^{-1}(x,y)$	8	coma	$3y^3+3x^2y-2y$		
$Z_3^3(x,y)$	9	trefoil	x^3-3xy^2		
$Z_3^{-3}(x,y)$	10	trefoil	$3x^2y-y^3$		
$Z_4^0(x,y)$	11	spherical	$6x^4+12y^2x^2-6x^2+6y^4-6y^2+1$		
$Z_4^2(x,y)$	12		$4x^4-3x^2-4y^4+3y^2$		
$Z_4^{-2}(x,y)$	13		$8yx^3+8y^3x-6yx$		
$Z_4^4(x,y)$	14		$x^4-6y^2x^2+y^4$		
$Z_4^{-4}(x,y)$	15		$4x^3y-4xy^3$		

Figure A-3: First Zernike polynomials in Cartesian coordinates. Source: [24]

A-1-7 Zernike-Kolmogorov

For Kolmogorov turbulence, as the mode number *J* increases, defined by Noll [19], the contribution of the J-th mode to the phase decreases [55].

$\Delta_1 = 1.0299(D/r_0)^{5/3}$	$\Delta_{12} = 0.0352(D/r_0)^{5/3}$
$\Delta_2 = 0.582(D/r_0)^{5/3}$	$\Delta_{13} = 0.0328(D/r_0)^{5/3}$
$\Delta_3 = 0.134(D/r_0)^{5/3}$	$\Delta_{14} = 0.0304(D/r_0)^{5/3}$
$\Delta_4 = 0.111(D/r_0)^{5/3}$	$\Delta_{15} = 0.0279(D/r_0)^{5/3}$
$\Delta_5 = 0.088(D/r_0)^{5/3}$	$\Delta_{16} = 0.0267(D/r_0)^{5/3}$
$\Delta_6 = 0.0648(D/r_0)^{5/3}$	$\Delta_{17} = 0.0255(D/r_0)^{5/3}$
$\Delta_7 = 0.0587(D/r_0)^{5/3}$	$\Delta_{18} = 0.0243(D/r_0)^{5/3}$
$\Delta_8 = 0.0525(D/r_0)^{5/3}$	$\Delta_{19} = 0.0232(D/r_0)^{5/3}$
$\Delta_9 = 0.0463(D/r_0)^{5/3}$	$\Delta_{20} = 0.0220(D/r_0)^{5/3}$
$\Delta_{10} = 0.0401(D/r_0)^{5/3}$	$\Delta_{21} = 0.0208(D/r_0)^{5/3}$
$\Delta_{11} = 0.0377(D/r_0)^{5/3}$	$\Delta_J \sim 0.2944J^{-\sqrt{3}/2}$

Figure A-4: Zernike-Kolmogorov mean square residual phase errors (Δ_J). D is the aperture size and r_0 is Frieds parameter.

Frieds parameter is the diameter of the sub-aperture where the RMS of the phase is statistically equal to 1 rad. The ratio of D/r_0 is an indication of the turbulence strength.

A-1-8 Charged Coupled Device

A Charged Coupled Device (CCD) is an photon detector and is divided into multiple small areas called pixels. Incident photons are proportionally converted to electrons. Exposure time is used such that the 'potential well' of each pixel does not exceeds its capacity. The resolution of a CCD is defined by the size of its pixels and their individual separation (pitch).

Several important properties of a CCD are: Quantum Efficiency, Wavelength range, Dynamic Range and Noise. *Quantum Efficiency (QE)* refers to the ratio photons hitting the CCD to conversion into electrical impuls. Not all incident photons are actually detected. The Best CCD's achieve a QE of 80%.
Wavelength range refers to the operation range of a CCD, ranging from 400 nm to about 1050 nm with a peak sensitivity at 700nm. *Dynamic Range* is the difference between the brightest and faintest source a detector can distinguish. Essentially its the minimum and maximum amount of electrons which can be collected by a pixel. The maximum value is determined by the pixel saturation value. The minimum value is determined by the CCD physical structure causing electronic noise.
Noise response is an important parameter and two of its main contributors are listed here:

- Dark current - noise generated through thermal heating. The lower the temperature of the CCD, the lower this noise generation becomes.
- Readout noise - generated through the conversion of electrons in each pixel to a voltage on the CCD output.

The photocurrent *i* generated through absorbed photons and its linearly proportional to the incident power (*P*):

$$\mathrm{i} = \mathrm{RP} \tag{A-9}$$

The constant R is the responsivity and is given by:

$$\mathcal{R} = \frac{\eta_{qe} q}{hv} \tag{A-10}$$

Where η_{qe} is the QE, *q* the electronic charge, *h* Planck's constant and ν the optical frequency. Thus, the measurable constant is the optical power which is related to the complex field equation (see Section A-2-2).

A-1-9 Wavefront Error

Wavefront error is defined as the difference between a diffraction limited wavefront and the measured wavefront. It is an important quantity defining the quality of an optical system.

A-2 Imaging Theory

A-2-1 Fourier Transform

Fourier methods can be used to describe an optical system, such as the Point Spread Function (PSF). The definition of the (two-dimensional) Fourier Transform of a complex-valued function g(x,y) is:

$$G\left(f_x, f_y\right) = \mathcal{F}\{g(x,y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\mathrm{e}^{-2\pi((f,x+f,r)}dxdy \tag{A-11}$$

with f_x and f_y being the spatial frequencies.

A-2-2 Monochromatic light and intensity

A monochromatic (single-frequency) wave propagating through an imaging system perpendicular to the *x*-, *y*-plane can be expressed at this location through equation [56]:

$$U\left(x_1, y_1\right) = A\left(x_1, y_1\right)\mathrm{e}^{i\phi(x_1,y_1)} \tag{A-12}$$

With amplitude *A* and phase ϕ. This field equation squared yields the intensity of the wave:

$$I\left(x_1, y_1\right) = U\left(x_1, y_1\right)U\left(x_1, y_1\right)^* = \left|U\left(x_1, y_1\right)\right|^2 \tag{A-13}$$

A-2-3 Fresnel-Kirchhoff Diffraction

The propagation of light within an optical system can be described using Fresnel-Kirchhoff diffraction theory. The Fresnel diffraction integral is expressed as

$$U(u,v) = \frac{z}{i\lambda} \iint_{\Sigma} U(x,y) \frac{\mathrm{e}^{ikr}}{r^2} dxdy \tag{A-14}$$

with $r = \sqrt{z^2 + (u-x)^2 + (v-y)^2}$.

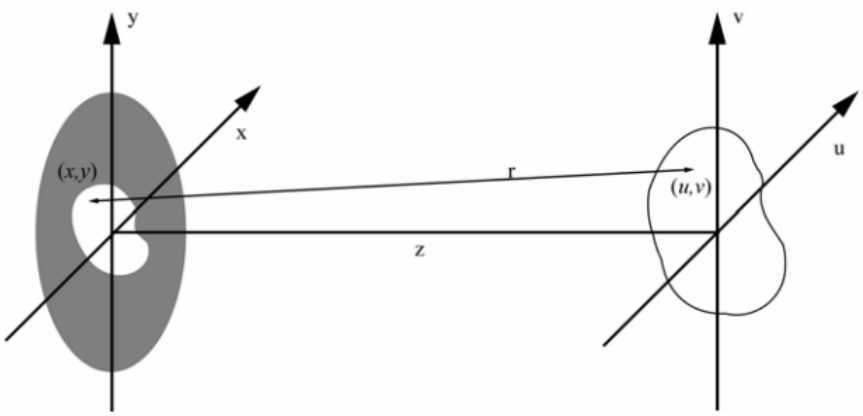


Figure A-5: Diffraction geometry according to Equation A-14 for pupil- and imaging plane.

After simplifications to the distance r, Equation A-14 can be rewritten to

$$U(u,v) = \frac{\mathrm{e}^{ikz}}{i\lambda z} \iint_{-\infty}^{\infty} U(x,y) \mathrm{e}^{\frac{ik}{2z}\left((u-x)^2+(v-y)^2\right)} dxdy \tag{A-15}$$

where $U(x,y)$ is zero outside the aperture. Algebraically manipulating this Equation yields:

$$U(u,v) = \frac{\mathrm{e}^{ik}}{i\lambda z} \mathrm{e}^{\frac{ik}{2z}(u^2+v^2)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U(x,y) \cdot \mathrm{e}^{\frac{ik}{2z}\left(x^2+y^2\right)} \cdot \mathrm{e}^{-\frac{2\pi i}{\lambda z}(ux+vu)} dxdy \tag{A-16}$$

which is the Fourier transform of a field within an aperture multiplied by a quadratic phase factor. For a more thorough analysis of Fresnel's diffraction integrals, the reader is referred to [56].

A-3 Deep Learning

A-3-1 Training, Validation and Testing

Deep learning models learn from and make predictions on data. The data set used to build a Deep Learning model, can be subdivided into three parts: a training set, a validation set and test set. The training set contains the majority of data and is used to fit the parameters during training. The validation set provides a unbiased evaluation of the model fit on the training set. Since the model has not seen the validation data before, the validation error will always be higher than the training error. Often, multiple parameters are tuned (e.g. amount of hidden layers, amount of

neurons or regularization intensity) such that multiple models emerge. The final model would be the model with the lowest validation error. The test set is used for an unbiased evaluation of the best achieving model.

Appendix B

CODE

B-1 Neural Networks

In this section the implementation of the neural networks in Python is shown. All networks have been implemented using Keras. Each line of code denotes a different layer in the network and is structured from input (top) to output (bottom). Each convolutional layers shows in order: amount of filters, kernel size, activation function and padding. Dense layers show the amount of neurons and activation function. Dropout layers indicate the dropout ratio. Note that all output layers end with a linear activation function, which enable regression.

B-1-1 Alex-Net

```
1
2  model = keras.models.Sequential([
3
4  keras.layers.Conv2D(32, 5, activation="relu", padding="same",
5                      input_shape=[256, 256, 1]),
6  keras.layers.MaxPooling2D(),
7  keras.layers.Conv2D(32, 5, activation="relu", padding="same"),
8  keras.layers.MaxPooling2D(),
9  keras.layers.Conv2D(64, 3, activation="relu", padding="same"),
10 keras.layers.Conv2D(64, 3, activation="relu", padding="same"),
11 keras.layers.Conv2D(64, 3, activation="relu", padding="same"),
12 keras.layers.MaxPooling2D(),
13 keras.layers.Flatten(),
14 keras.layers.Dense(512, activation="relu"),
15 keras.layers.Dropout(0.25),
16 keras.layers.Dense(512, activation="relu"),
17 keras.layers.Dropout(0.25),
18 keras.layers.Dense(31, activation="linear")
19
20 ])
```

B-1-2 U-Net

```
1
2  inputs = keras.Input((256, 256, 1))
3
4  c1 = keras.layers.Conv2D(8, (3, 3),activation='relu', padding='same') (
       inputs)
5  c1 = keras.layers.Conv2D(8, (3, 3),activation='relu', padding='same') (c1
       )
6  p1 = keras.layers.MaxPooling2D((2, 2)) (c1)
7
8  c2 = keras.layers.Conv2D(16, (3, 3),activation='relu', padding='same') (
       p1)
9  c2 = keras.layers.Conv2D(16, (3, 3),activation='relu', padding='same') (
       c2)
10 p2 = keras.layers.MaxPooling2D((2, 2)) (c2)
11
12 c3 = keras.layers.Conv2D(32, (3, 3),activation='relu', padding='same') (
       p2)
13 c3 = keras.layers.Conv2D(32, (3, 3),activation='relu', padding='same') (
       c3)
14 p3 = keras.layers.MaxPooling2D((2, 2)) (c3)
15
16 c4 = keras.layers.Conv2D(64, (3, 3),activation='relu', padding='same') (
       p3)
17 c4 = keras.layers.Conv2D(64, (3, 3),activation='relu', padding='same') (
       c4)
18 p4 = keras.layers.MaxPooling2D(pool_size=(2, 2)) (c4)
19
20 c5 = keras.layers.Conv2D(128, (3, 3),activation='relu', padding='same') (
       p4)
21 c5 = keras.layers.Conv2D(128, (3, 3),activation='relu', padding='same') (
       c5)
22
23 u6 = keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='
       same') (c5)
24 u6 = keras.layers.concatenate([u6, c4])
25 c6 = keras.layers.Conv2D(64, (3, 3),activation='relu', padding='same') (
       u6)
26 c6 = keras.layers.Conv2D(64, (3, 3),activation='relu', padding='same') (
       c6)
27
28 u7 = keras.layers.Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='
       same') (c6)
29 u7 = keras.layers.concatenate([u7, c3])
30 c7 = keras.layers.Conv2D(32, (3, 3),activation='relu', padding='same') (
       u7)
31 c7 = keras.layers.Conv2D(32, (3, 3),activation='relu', padding='same') (
       c7)
32
33 u8 = keras.layers.Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='
       same') (c7)
34 u8 = keras.layers.concatenate([u8, c2])
```

```
35 c8 = keras.layers.Conv2D(16, (3, 3),activation='relu', padding='same') (
       u8)
36 c8 = keras.layers.Conv2D(16, (3, 3),activation='relu', padding='same') (
       c8)
37
38 u9 = keras.layers.Conv2DTranspose(8, (2, 2), strides=(2, 2), padding='
       same') (c8)
39 u9 = keras.layers.concatenate([u9, c1], axis=3)
40 c9 = keras.layers.Conv2D(8, (3, 3),activation='relu', padding='same') (u9
       )
41 c9 = keras.layers.Conv2D(8, (3, 3),activation='relu', padding='same') (c9
       )
42
43 outputs = keras.layers.Conv2D(1, (1, 1), activation='linear') (c9)
```

B-1-3 Xception

```
1
2  input_shape = _obtain_input_shape(input_shape,
3                                    default_size=299,
4                                    min_size=71,
5                                    data_format=K.image_data_format(),
6                                    include_top=include_top)
7
8
9      img_input = Input(shape=input_shape)(x)
10     x = Conv2D(32, (3, 3), strides=(2, 2), use_bias=False, name='
           block1_conv1')(img_input)
11     x = BatchNormalization(name='block1_conv1_bn')(x)
12     x = Activation('relu', name='block1_conv1_act')(x)
13     x = Conv2D(64, (3, 3), use_bias=False, name='block1_conv2')(x)
14     x = BatchNormalization(name='block1_conv2_bn')(x)
15     x = Activation('relu', name='block1_conv2_act')(x)
16
17     residual = Conv2D(128, (1, 1), strides=(2, 2),
18                       padding='same', use_bias=False)(x)
19     residual = BatchNormalization()(residual)
20
21     x = SeparableConv2D(128, (3, 3), padding='same', use_bias=False, name
           ='block2_sepconv1')(x)
22     x = BatchNormalization(name='block2_sepconv1_bn')(x)
23     x = Activation('relu', name='block2_sepconv2_act')(x)
24     x = SeparableConv2D(128, (3, 3), padding='same', use_bias=False, name
           ='block2_sepconv2')(x)
25     x = BatchNormalization(name='block2_sepconv2_bn')(x)
26
27     x = MaxPooling2D((3, 3), strides=(2, 2), padding='same', name='
           block2_pool')(x)
28     x = layers.add([x, residual])
29
30     residual = Conv2D(256, (1, 1), strides=(2, 2),
31                       padding='same', use_bias=False)(x)
```

```
32      residual = BatchNormalization()(residual)
33
34      x = Activation('relu', name='block3_sepconv1_act')(x)
35      x = SeparableConv2D(256, (3, 3), padding='same', use_bias=False, name
            ='block3_sepconv1')(x)
36      x = BatchNormalization(name='block3_sepconv1_bn')(x)
37      x = Activation('relu', name='block3_sepconv2_act')(x)
38      x = SeparableConv2D(256, (3, 3), padding='same', use_bias=False, name
            ='block3_sepconv2')(x)
39      x = BatchNormalization(name='block3_sepconv2_bn')(x)
40
41      x = MaxPooling2D((3, 3), strides=(2, 2), padding='same', name='
            block3_pool')(x)
42      x = layers.add([x, residual])
43
44      residual = Conv2D(728, (1, 1), strides=(2, 2),
45                        padding='same', use_bias=False)(x)
46      residual = BatchNormalization()(residual)
47
48      x = Activation('relu', name='block4_sepconv1_act')(x)
49      x = SeparableConv2D(728, (3, 3), padding='same', use_bias=False, name
            ='block4_sepconv1')(x)
50      x = BatchNormalization(name='block4_sepconv1_bn')(x)
51      x = Activation('relu', name='block4_sepconv2_act')(x)
52      x = SeparableConv2D(728, (3, 3), padding='same', use_bias=False, name
            ='block4_sepconv2')(x)
53      x = BatchNormalization(name='block4_sepconv2_bn')(x)
54
55      x = MaxPooling2D((3, 3), strides=(2, 2), padding='same', name='
            block4_pool')(x)
56      x = layers.add([x, residual])
57
58      for i in range(8):
59          residual = x
60          prefix = 'block' + str(i + 5)
61
62          x = Activation('relu', name=prefix + '_sepconv1_act')(x)
63          x = SeparableConv2D(728, (3, 3), padding='same', use_bias=False,
                name=prefix + '_sepconv1')(x)
64          x = BatchNormalization(name=prefix + '_sepconv1_bn')(x)
65          x = Activation('relu', name=prefix + '_sepconv2_act')(x)
66          x = SeparableConv2D(728, (3, 3), padding='same', use_bias=False,
                name=prefix + '_sepconv2')(x)
67          x = BatchNormalization(name=prefix + '_sepconv2_bn')(x)
68          x = Activation('relu', name=prefix + '_sepconv3_act')(x)
69          x = SeparableConv2D(728, (3, 3), padding='same', use_bias=False,
                name=prefix + '_sepconv3')(x)
70          x = BatchNormalization(name=prefix + '_sepconv3_bn')(x)
71
72          x = layers.add([x, residual])
73
74      residual = Conv2D(1024, (1, 1), strides=(2, 2),
75                        padding='same', use_bias=False)(x)
```

```
76      residual = BatchNormalization()(residual)
77
78      x = Activation('relu', name='block13_sepconv1_act')(x)
79      x = SeparableConv2D(728, (3, 3), padding='same', use_bias=False, name
           ='block13_sepconv1')(x)
80      x = BatchNormalization(name='block13_sepconv1_bn')(x)
81      x = Activation('relu', name='block13_sepconv2_act')(x)
82      x = SeparableConv2D(1024, (3, 3), padding='same', use_bias=False,
           name='block13_sepconv2')(x)
83      x = BatchNormalization(name='block13_sepconv2_bn')(x)
84
85      x = MaxPooling2D((3, 3), strides=(2, 2), padding='same', name='
           block13_pool')(x)
86      x = layers.add([x, residual])
87
88      x = SeparableConv2D(1536, (3, 3), padding='same', use_bias=False,
           name='block14_sepconv1')(x)
89      x = BatchNormalization(name='block14_sepconv1_bn')(x)
90      x = Activation('relu', name='block14_sepconv1_act')(x)
91
92      x = SeparableConv2D(2048, (3, 3), padding='same', use_bias=False,
           name='block14_sepconv2')(x)
93      x = BatchNormalization(name='block14_sepconv2_bn')(x)
94      x = Activation('relu', name='block14_sepconv2_act')(x)
95
96
97      x = GlobalAveragePooling2D(name='avg_pool')(x)
98
99       x = Dropout(0.5)(x)
100     predictions = Dense(31, activation='linear')(x)
```

Bibliography

[1] "Neptune from the vlt with and without adaptive optics." https://www.eso.org/public/images/eso1824b/. Accessed: 2020-04-29.

[2] J. P. Hope Queener, *Adaptive Optics for Vision Science*. 2005.

[3] "An introduction to neural networks." https://www.inf.ed.ac.uk/teaching/courses/nlu/assets/reading/Gurney_et_al.pdf. Accessed: 2019-12-07.

[4] "Convolutional neural networks for text classification." http://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/. Accessed: 2019-07-07.

[5] "What is max pooling in convolutional neural networks?." https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks. Accessed: 2019-07-07.

[6] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: a contemporary overview," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 87–109, 2015.

[7] M. C. Joel Kubby, Sylvain Gigan, *Wavefront Shaping for Biomedical Imaging*. Cambridge University Press, 2019.

[8] D. L. Fried, "Least-square fitting a wave-front distortion estimate to an array of phase-difference measurements," *JOSA*, vol. 67, no. 3, pp. 370–375, 1977.

[9] T. K. Barrett and D. G. Sandler, "Artificial neural network for the determination of hubble space telescope aberration from stellar images," *Applied optics*, vol. 32, no. 10, pp. 1720–1727, 1993.

[10] D. A. Montera, B. M. Welsh, M. C. Roggemann, and D. W. Ruck, "Processing wave-front-sensor slope measurements using artificial neural networks," *Applied optics*, vol. 35, no. 21, pp. 4238–4251, 1996.

[11] Y. Xu, D. He, Q. Wang, H. Guo, Q. Li, Z. Xie, and Y. Huang, "An improved method of measuring wavefront aberration based on image with machine learning in free space optical communication," *Sensors*, vol. 19, no. 17, p. 3665, 2019.

[12] H. Guo, Y. Xu, Q. Li, S. Du, D. He, Q. Wang, and Y. Huang, "Improved machine learning approach for wavefront sensing," *Sensors*, vol. 19, no. 16, p. 3533, 2019.

[13] Y. Nishizaki, M. Valdivia, R. Horisaki, K. Kitaguchi, M. Saito, J. Tanida, and E. Vera, "Deep learning wavefront sensing," *Optics express*, vol. 27, no. 1, pp. 240–251, 2019.

[14] R. Swanson, M. Lamb, C. Correia, S. Sivanandam, and K. Kutulakos, "Wavefront reconstruction and prediction with convolutional neural networks," in *Adaptive Optics Systems VI*, vol. 10703, p. 107031F, International Society for Optics and Photonics, 2018.

[15] H. Guo, N. Korablinova, Q. Ren, and J. Bille, "Wavefront reconstruction with artificial neural networks," *Optics express*, vol. 14, no. 14, pp. 6456–6462, 2006.

[16] C. Max, "Introduction to adaptive optics and its history," in *American Astronomical Society 197th Meeting*, vol. 30, pp. 658407–1, NSF Center for Adaptive Optics University of California at Santa Cruz and . . . , 2001.

[17] "Adaptive optics – the next generation. 11,000 actuators to aid search for exoplanets with the e-elt." https://www.pi-usa.us/en/tech-blog/. Accessed: 2019-11-06.

[18] B. C. Platt and R. Shack, "History and principles of shack-hartmann wavefront sensing," *Journal of refractive surgery*, vol. 17, no. 5, pp. S573–S577, 2001.

[19] R. J. Noll, "Zernike polynomials and atmospheric turbulence," *JOsA*, vol. 66, no. 3, pp. 207–211, 1976.

[20] "Wavefront sensing of light beams." http://www.iitg.ac.in/physics/fac/brboruah/htmls/wfs.html. Accessed: 2019-10-03.

[21] "Metrology and sensing." https://www.iap.uni-jena.de/iapmedia/de/Lecture/Optical+Metrology+and+Sensing1519858800/OMS17_Metrology+and+Sensing+Lecture+7+Wavefront+sensors-p-20002855.pdf. Accessed: 2020-05-03.

[22] "Shack-hartmann wavefront sensors." https://www.thorlabs.com/catalogpages/V21/1613.PDF. Accessed: 2020-03-11.

[23] R. W. Gerchberg, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, vol. 35, pp. 237–246, 1972.

[24] O. S. G. V. Michel Verhaegen, Paulo Pozzi and D. Wilding, "Control for high resolution imaging," *Lecture notes for the course sc42030*, pp. 61–71, 2017.

[25] L. M. Mugnier, A. Blanc, and J. Idier, "Phase diversity: a technique for wave-front sensing and for diffraction-limited imaging," *Advances in Imaging and Electron Physics*, vol. 141, pp. 1–76, 2006.

[26] D. R. Gerwe, M. M. Johnson, and B. Calef, "Local minima analysis of phase diverse phase retrieval using maximum likelihood," in *The Advanced Maui Optical and Space Surveillance Technical Conference*, 2008.

[27] B. L. Ellerbroek and C. R. Vogel, "Inverse problems in astronomical adaptive optics," *Inverse Problems*, vol. 25, no. 6, p. 063001, 2009.

[28] J. R. P. Angel, P. Wizinowich, M. Lloyd-Hart, and D. Sandler, "Adaptive optics for array telescopes using neural-network techniques," *Nature*, vol. 348, no. 6298, p. 221, 1990.

[29] Z. Li and X. Li, "Centroid computation for shack-hartmann wavefront sensor in extreme situations based on artificial neural networks," *Optics express*, vol. 26, no. 24, pp. 31675–31692, 2018.

[30] Y. Bengio, A. Courville, and I. J. Goodfellow, "Deep learning: adaptive computation and machine learning," *Bengio. A. Courville*, 2016.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[32] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[33] "Cyclical learning rates for training neural networks." https://mc.ai/cyclical-learning-rates-for-training-neural-networks/. Accessed: 2020-04-29.

[34] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization. international conference on learning representations (2015)," 2015.

[35] Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for nonconvex optimization," *corr abs/1502.04390*, 2015.

[36] "Convolutional neural network (cnn)." https://www.tensorflow.org/tutorials/images/cnn. Accessed: 2019-05-07.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[39] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010.

[40] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[41] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition. arxiv preprint arxiv: 1512.03385," 2015.

[44] "Depth wise separable convolutional neural networks." https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks. Accessed: 2019-10-31.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[46] M. Lamb, D. R. Andersen, J.-P. Véran, C. Correia, G. Herriot, M. Rosensteiner, and J. Fiege, "Non-common path aberration corrections for current and future ao systems," in *Adaptive Optics Systems IV*, vol. 9148, p. 914857, International Society for Optics and Photonics, 2014.

[47] R. R. Shannon, *Handbook of Optics*. 1994.

[48] M. Townson, O. Farley, G. O. de Xivry, J. Osborn, and A. Reeves, "Aotools: a python package for adaptive optics modelling and analysis," *Optics express*, vol. 27, no. 22, pp. 31316–31329, 2019.

[49] "Ccd noise sources and signal-to-noise ratio." https://hamamatsu.magnet.fsu.edu/articles/ccdsnr.html. Accessed: 2019-10-30.

[50] J. Schmidt, "Numerical simulation of optical wave propagation with examples in matlab," Society of Photo-Optical Instrumentation Engineers, 2010.

[51] "Keras: the python deep learing api." https://keras.io/. Accessed: 2019-11-10.

[52] V. Lakshminarayanan and A. Fleck, "Zernike polynomials: a guide," *Journal of Modern Optics*, vol. 58, no. 7, pp. 545–561, 2011.

[53] Y. Jin, Y. Zhang, L. Hu, H. Huang, Q. Xu, X. Zhu, L. Huang, Y. Zheng, H.-L. Shen, W. Gong, *et al.*, "Machine learning guided rapid focusing with sensor-less aberration corrections," *Optics express*, vol. 26, no. 23, pp. 30162–30171, 2018.

[54] J. Antonello, "Optimisation-based wavefront sensorless adaptive optics for microscopy," 2014.

[55] A. N. Kolmogorov, "A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high reynolds number," *Journal of Fluid Mechanics*, vol. 13, no. 1, pp. 82–85, 1962.

[56] J. W. Goodman, *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.